VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Diplomová práce

2011 Bc. Richard Lapiš

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Informační systém pro správu hostingu přes webové rozhraní

Web interface hosting control information system

2011 Bc. Richard Lapiš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Richard Lapiš**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Informační systém pro správu hostingu přes webové rozhraní
Web Interface Hosting Control Information System

Zásady pro vypracování:

Diplomant má za úkol vytvoření webového informačního systému pro vzdálenou správu hostingového serveru unixového typu (Linux). Aplikace se bude skládat ze dvou částí, které spolu budou spolupracovat - webové rozhraní pro správu serveru a samotná aplikace pro automatické řízení serveru.
1. Najděte existující aplikace pro (vzdálenou) správu serveru, prostudujte je a porovnejte jejich vlastnosti.
2. Zjistěte požadavky na IS pro vzdálenou správu hostingového serveru (správa uživatelů a jejich účtů, správa domén, správa FTP (FTPS) serveru, správa databází uživatelů (MySQL nebo SQLite), generování statistik přístupů, správa Cronu, SVN server, systém zálohování a obnovy dat na serveru, správa plateb, OpenVPN (+ certifikáty), veškeré logování činnosti serveru a pro automatické řízení serveru).
3. Vytvořte analýzu IS podle zjištěných požadavků a s ohledem na bezpečnost aplikace.
4. Zpracujte návrh implementace pro vybrané vhodné SW a HW prostředky vhodné pro vzdálenou správu serveru.
5. Navržený IS implementujte a otestujte, porovnejte s existujícími aplikacemi.
6. Vypracujte programátorské a uživatelské příručky pro jednotlivé role v IS.
7. Vypracujte aplikaci pro automatické řízení serveru a otestujte.

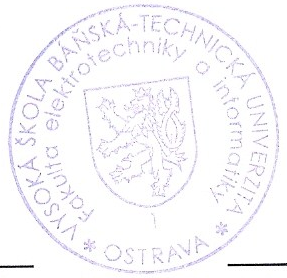
Seznam doporučené odborné literatury:

ŠARMANOVÁ, Jana. *Teorie zpracování dat.* Ostrava: VŠB TU Ostrava, 1997. 108 s. ISBN 80-7078-491-1.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Emilie Šeptáková**

Datum zadání: 19.11.2010
Datum odevzdání: 06.05.2011



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 6.5.2011 v Ostravě.

……..

Podpis

Poděkování

Rád bych poděkoval vedoucí diplomové práce Ing. Emilii Šeptákové za odbornou pomoc a konzultaci při vytváření této práce.

Abstrakt

Cílem této diplomové práce je vyvinout webovou aplikaci, resp. informační systém pro správu hostingového serveru. Aplikace se bude skládat ze dvou částí. První část bude webová, která bude umístěna na internetu a uživatelé k ní budou přistupovat prostřednictvím internetového prohlížeče. Druhá část bude aplikace pro automatické řízení serveru. Obě tyto části spolu budou komunikovat. Obsluhovat budou veškerou důležitou funkčnost serveru, který je postaven na Unixovém základu (Linux). Mezi ty nejdůležitější funkce se řadí např. správa FTP serverů, SVN serverů, databází, domén a uživatelských účtů, Cron úloh, systém zálohování a obnovy dat serveru aj. Výsledná webová aplikace bude volně šiřitelným řešením pro servery se stejnou konfigurací, pro kterou je vyvíjena.

Klíčová slova

Server, informační systém, hosting, Ruby on rails, SVN, FTP, Doména, MySQL, Záloha, Cron, Linux

Abstract

The goal of this thesis is to develop a web application (in other words information system) for hosting server management. The application will consist of two parts. The first part will be a web part, which will be placed on internet, where the users can access it by browser. The second part will be an application automatically managing the server. Both of these parts will communicate with each other. They'll secure all the important functions of the Unix-based server (Linux). The most important functions include for example management of FTP servers, SVN servers, databases, domains and user accounts, Cron tasks, backup and data recovery systems and others. The final web application will be open source solution for servers with same configuration as the one, for which it was developed.

Key words

Server, information system, hosting, Ruby on rails, SVN, FTP, Domain, MySQL, Backup, Cron, Linux

Seznam použitých symbolů a zkratek

BSD	Berkeley Software Distribution
Cron	systém pro automatické spouštění procesů
CVS	Concurrent Version System – verzovací systém souborů
DFD	Data Flow Diagram
DNS	Domain Name System – hierarchický systém doménových jmen
FTP	File Transfer Protocol – protokol pro přenos dat v počítačové síti
GPL	General Public License – licence pro svobodný software
IDE	Integrated Development Environment – vývojové prostředí
LDAP	Lightweight Directory Access Protocol
MIME	Multipurpose Internet Mail Extensions
MVC	Model View Controller – třívrstvá architektura
PDF	Portable Document Format
RAID	Redundant Array of Independent Disks – metoda ochrany pevných disků
REST	Representational State Transfer
RUBY	interpretovaný skriptovací programovací jazyk
SOAP	Simple Object Access Protocol – protokol pro vzdálené volání procedur
SŘBD	systém řízení báze dat
SSH	Secure Shell
SSL	Secure Sockets Layer – bezpečnostní protokol
SVN	subversion system – systém pro správu a verzování zdrojových kódů
UNIX	základ operačního systému
URI	Uniform Resource Identifier – jednoznačný identifikátor zdroje
XML-RPC	XML Remote Procedure Call – vzdálené volání procedur

Seznam tabulek

Seznam obrázků

Seznam příkladů

Obsah

1 Úvod

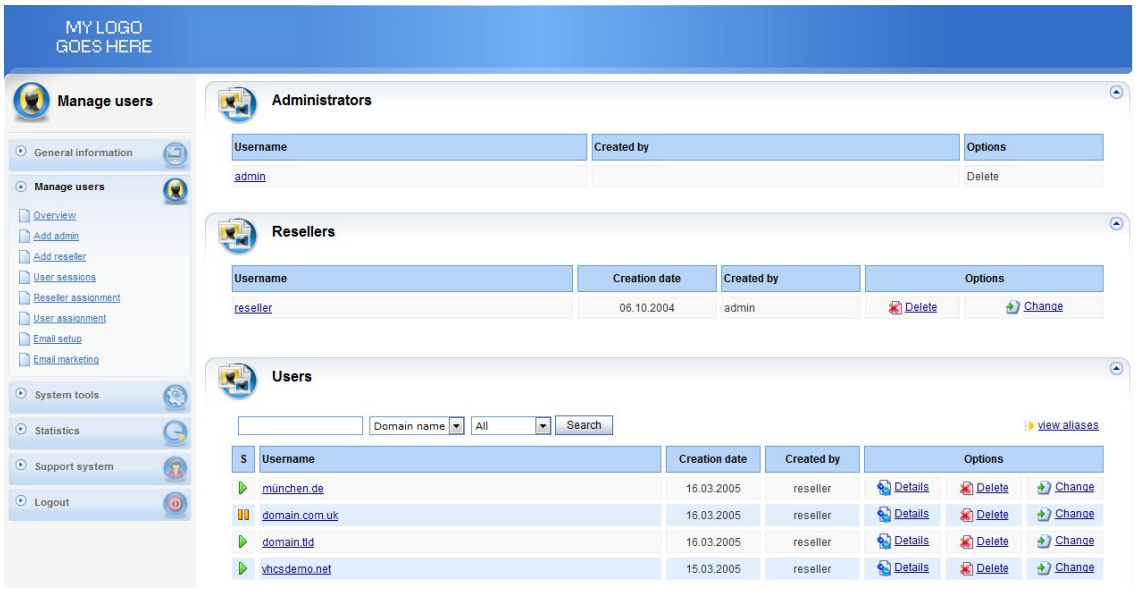
Tato diplomová práce má za úkol vyvinout webovou aplikaci pro vzdálenou správu hostingového serveru. Aplikace bude tvořena na míru Unixového serveru. V první kapitole srovnám výhody a nevýhody již hotových řešení pro vzdálenou správu hostingu, které jsou dostupné na internetu. Následně seznámím čtenáře se zadáním práce, požadavky na systém a s nimi spojenou problematikou serverů. V pozdějších kapitolách se budu zabývat analýzou a návrhem těchto požadavků. Čtenář bude také seznámen s použitými technologiemi, ať už s implementačním jazykem, tak např. třívrstvou softwarovou architekturou MVC. Poslední kapitoly se budou již věnovat samotnému řešení aplikace, a to jak webové části, tak i automatizované serverové, což je obsahem implementace. Na závěr této práce shodnotím dosažené výsledky a nasazení tohoto systému v praxi.

2 Současná řešení

Na internetu je mnoho řešení pro správu webhostingových serverů prostřednictvím webového prohlížeče, některá jsou velmi propracovaná, některá méně. Profesionální řešení, které nabízí služby nad rámec obyčejných systémů jsou velmi kvalitní, ale jsou většinou komerční a tudíž za poplatek. V této části se budu věnovat srovnání těch řešení, která jsou volně dostupná (open source) pod licencí nejčastěji GPL a jsou určená pro servery postavené na systému Linux. Srovnám nejznámější a nejrozšířenější z nich, a to jsou Virtual Hosting Control System, OpenPanel, Webmin a ISPConfig.

2.1 Virtual Hosting Control System

Virtual Hosting Control System (dále jen VHCS) je volně dostupný produkt pro webovou správu hostingových služeb. Poskytuje ho zahraniční firma moleSoftware GmbH na svých stránkách[1], kde si lze vyzkoušet i online demo. Jako implementační jazyky byly zvoleny PHP, PERL a C. V PHP je napsáno uživatelské rozhraní s kterým koncový uživatel pracuje, jazyk PERL byl použit pro VHCS Engine, který vše řídí a konečně v jazyku C byl napsán VHCS Daemon. Systém má tři typy uživatelů. Prodejce (Reseller), Administrátora a Koncového uživatele (Enduser).



Obrázek 1: VHCS uživatelské rozhraní

Uživatelské rozhraní lze vidět na obrázku 1, kde je přepnuta volba správy uživatelů, administrátorů a prodejců. Jedná se o obrazovku přihlášeného uživatele s právy Administrator.

[1] http://www.vhcs.net
[2] http://www.openpanel.com

2.1.1 Funkce systému

Systém VHCS má mnoho funkcí, vypíši ale jen ty nejdůležitější, které jsou funkcionálně nejpodstatnější pro správu serveru.

2.1.1.1 Práva Administrátora

- Plná správa uživatelů
- Správa nastavení serveru
- Statistiky serveru
- Správa Action logu a Server logu
- Správa SSL certifikátů
- Správa IP adres
- Správa podpory systému

2.1.1.2 Práva Prodejce (Reseller)

- Správa Koncových uživatelů
- Správa FTP účtů všech uživatelů
- Správa MySQL databází všech uživatelů
- Definice omezení na provoz
- Zálohy
- Webmail
- Správa domén, subdomén a doménových aliasů všech uživatelů
- Správa email účtů

2.1.1.3 Práva Koncového uživatele

- Správa svých domén, subdomén a doménových aliasů
- Správa POP3 účtů
- Webmail
- Správa svých FTP účtů
- Správa svých MySQL databází
- Statistiky
- Zálohy

2.1.2 Výhody a nevýhody systému

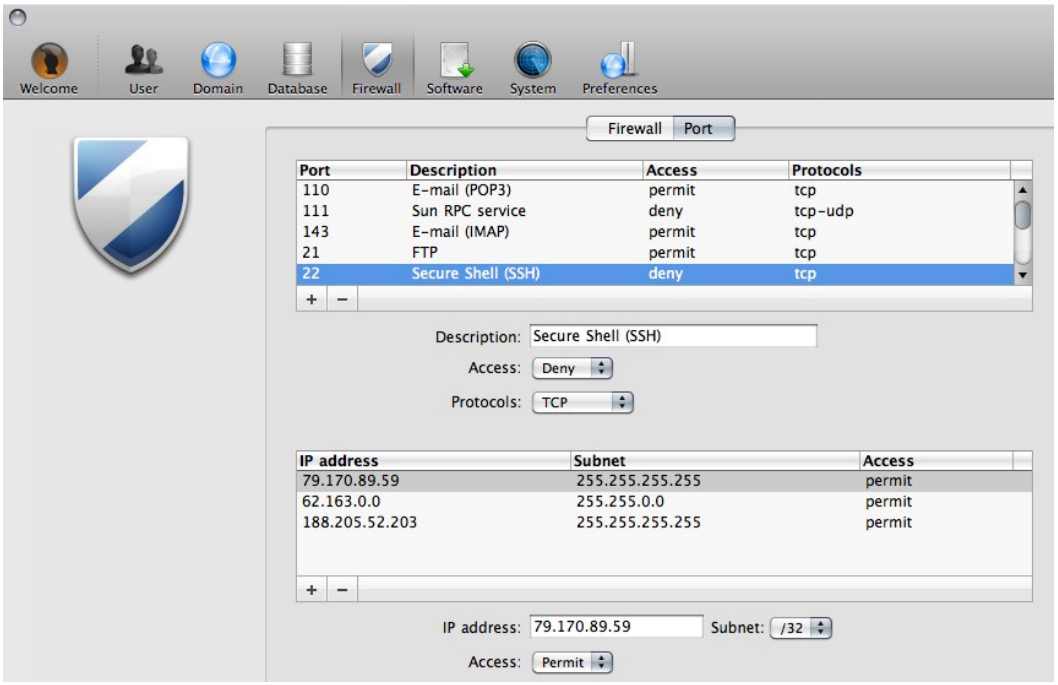
VHCS je vyvíjen již řadu let, a proto ho tvoří široká komunita uživatelů po celém světě. Má dobře vybudovanou podporu pro zákazníky a uživatelské rozhraní je dostupné v mnoha světových jazycích, včetně češtiny. Systém je po grafické stránce velmi dobře navržen.

Aplikace je rozdělena na tři různá rozhraní, pro každého z typů uživatelů zvlášť. Administrátor má právo pouze přidávat uživatele a spravovat chod serveru, případně sledovat různé vygenerované statistiky. Nemůže však spravovat další data uživatelů, jako jsou domény, FTP účty apod. K tomu má pří-

stup až Prodejce. Domnívám se, že i hlavní Administrátor by měl mít k těmto údajům přístup a možnost je případně upravit. Systému chybí správa Cron úloh.

2.2 OpenPanel

OpenPanel je vyvíjen skupinou vývojářů od počátku 90. let minulého století. Systém je volně ke stáhnutí na internetu[2], demo ukázka však zde k dispozici není. Ukázku rozhraní lze vidět na obrázku 2. Aplikace je momentálně dostupná ve verzi 3.0 a podporuje většinu Linuxových distribucí, jako např. Ubuntu, SuSe Linux, Debian.



Obrázek 2: OpenPanel uživatelské rozhraní

2.2.1 Funkce systému

OpenPanel obsahuje tyto hlavní funkce:

- Správu uživatelů
- Správu domén, subdomén
- Nastavení firewallu
- Správu FTP účtů
- Správa databází MySQL a PostgreSQL
- Správa serveru a jeho nastavení

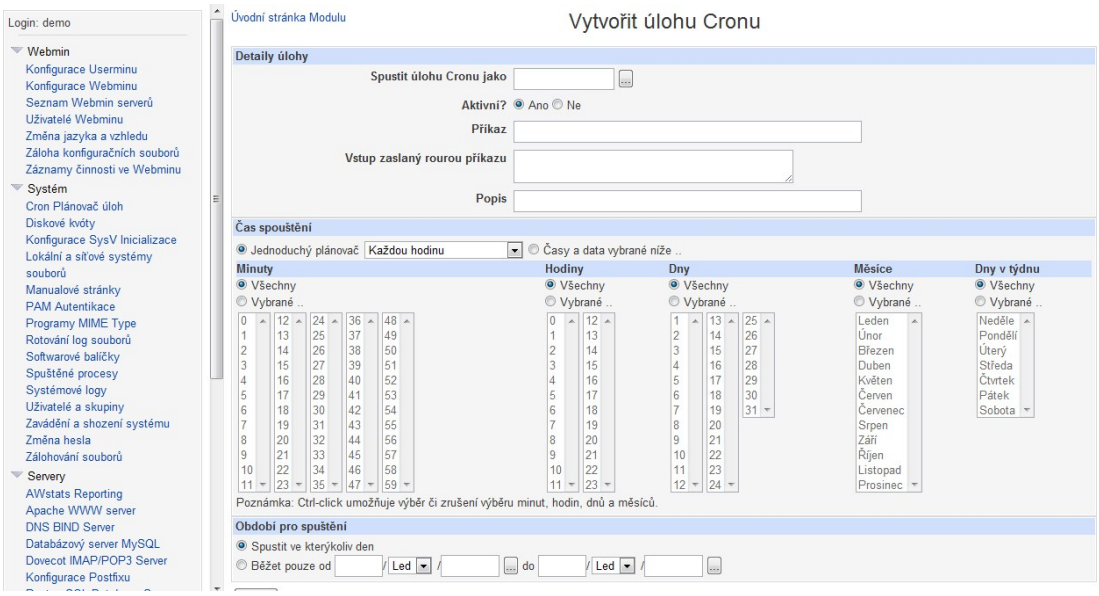
[2] http://www.openpanel.com

2.2.2 Výhody a nevýhody systému

OpenPanel je oproti předchozímu VHCS mnohem jednodušší a poskytuje také méně funkcí. Obsahuje ale ty nejdůležitější funkce, které potřebujeme my jako uživatelé se serverem provádět. Systému opět chybí dle mého názoru podstatná funkce vytváření Cron úloh a také doménových aliasů. Vůbec nebyla zpracována správa záloh. Výhodou oproti ostatním systémům je, že obsahuje OpenPanel-CLI, což je rozhraní s příkazovou řádkou pro uživatele, kteří si radši server spravují tímto způsobem. V příkazovém řádku lze provádět všechny operace jako v grafickém rozhraní. OpenPanel podporuje databázi PostgreSQL a dále lze přes jeho správu nastavovat i firewall serveru. Obsahuje také správu plateb za služby.

2.3 Webmin

Tuto webovou aplikaci vyvinul Jamie Cameron, který pracuje jako správce sítě ve společnosti Google. Je dostupná na internetu[3], kde si lze vyzkoušet i demo ukázku. Webmin je dostupný v mnoha jazycích, včetně češtiny a je kompatibilní s většinou Linuxových distribucí, ale i BSD systémům, Solaris, nebo také Windows. Možno je stáhnout základní aplikaci, která nabízí obvyklé funkce, ale lze dodat i rozšiřující balíčky. Je to jen na správci systému, jaké balíčky doinstaluje. Ukázkovou obrazovku aplikace lze vidět na obrázku 3.



Obrázek 3: Webmin uživatelské rozhraní

[3] http://www.webmin.com

2.3.1 Funkce systému

Webmin obsahuje jak základní funkce, které nalezneme i v konkurenčních systémech, tak i funkce, které má navíc. Mezi ty základní patří jmenovitě tyto:

- Správa uživatelů
- Správa MySQL a PostgreSQL databází
- Správa FTP účtů
- Správa domén, subdomén
- Zálohy
- Diskové kvóty

Mezi rozšířené funkce, které nejsou u těchto systémů běžné, bych zařadil:

- Nastavení DHCP serveru
- Nastavení LDAP
- Správa Cron úloh
- CVS Server
- Jabber IM Server
- SSH
- Doplňková instalace softwarových balíčků
- Rozšířené možnosti nastavení hardwaru serveru (partition, ...)

2.3.2 Výhody a nevýhody systému

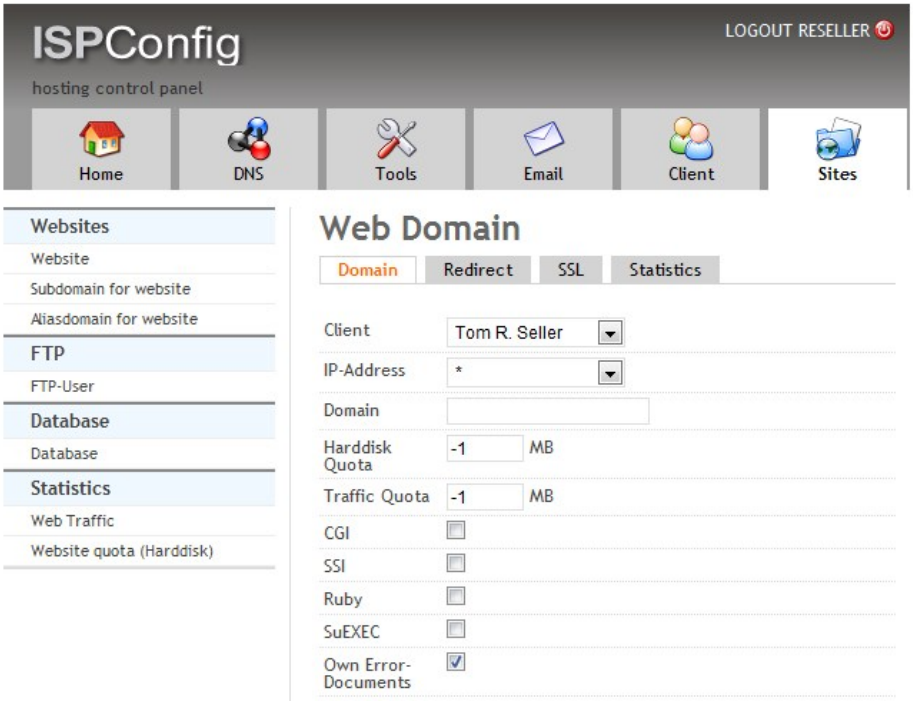
Výhoda tohoto systému spočívá v jeho jednoduchém a přehledném uživatelském rozhraní, které se opravdu svižně ovládá. Má spoustu funkcí, které bychom v konkurenčních systémech těžko hledali a ze srovnávaných systémů, které jsou nabízeny zdarma, má tuto sbírku funkcí největší. Co se týče skupin uživatelů, v systému nejsou. Při vytváření nového uživatele definujeme pouze bloky a sekce, ke kterým bude mít přístup a bude je moci ovládat. Toto řešení je velmi výhodné v případě, kdybychom chtěli některému z uživatelů později dodatečně některou z pokročilých funkcí zapnout, ale na druhou stranu nás to může zdržovat, když neustále každému uživateli budeme vybírat z tolika funkcí ty, které bude využívat. Tento systém již obsahuje pokročilou správu Cron úloh, která předchozím systémům chyběla. Každý uživatel si může sám na server pomocí tohoto rozhraní doinstalovat své softwarové balíčky, se kterými chce pracovat.

2.4 ISPConfig

Tato webová aplikace pro správu hostingového serveru je dostupná zdarma na internetu[4], kde si ji lze opět vyzkoušet online. Vyvíjí ji firma Projektfarm GmbH sídlící v Německu. S příchodem nové verze

[4] http://www.ispconfig.org

3.0 přibyla i podpora nových operačních systémů, jako je Open SuSE 11.4 apod. Ukázka uživatelského rozhraní je na obrázku 4.



Obrázek 4: ISPConfig uživatelské rozhraní

2.4.1 Funkce systému

Aplikace je rozdělena do třech sekcí, dostupných pro různé typy uživatelů, jako je tomu u systému VHCS. Jednotlivé skupiny uživatelů a jejich práva lze vidět v seznamu níže.

2.4.1.1 Práva Administrátora

- Plná správa uživatelů
- Správa FTP účtů všech uživatelů
- Správa domén, subdomén a doménových aliasů všech uživatelů
- Správa nastavení serveru
- Statistiky serveru
- Pokročilé sledování parametrů serveru a jeho logování
- Mailbox
- Zálohy
- Správa Cron úloh

2.4.1.2 Práva Prodejce (Reseller)

- Správa Koncových uživatelů

- Správa FTP účtů všech uživatelů
- Správa MySQL databází všech uživatelů
- Mailbox
- Správa domén, subdomén a doménových aliasů všech uživatelů

2.4.1.3 Práva Koncového uživatele (Client)

- Správa svých domén, subdomén a doménových aliasů
- Mailbox
- Správa svých FTP účtů
- Správa svých MySQL databází
- Statistiky

2.4.2 Výhody a nevýhody systému

ISPConfig je velmi dobře rozvržen mezi typy uživatelů. Je zpracován obdobně jako systém VHCS, ale administrátor zde již má přístup k funkcím jako jsou správa databází a FTP jednotlivých účtů apod. Na druhou stranu má mnohem méně funkcí než VHCS. Cron úlohy lze vytvářet, ale tato práva má pouze administrátor, což naopak vidím jako velkou nevýhodu tohoto systému, protože uživatel bude muset vždy kontaktovat administrátora, když bude chtít udělat zápis do Cron tabulky. Podobně je tomu u obnovení záloh, kterou si také uživatel nemůže provést sám, přístup k tomu má opět administrátor. ISPConfig je graficky přívětivější než předchozí systém Webmin. Systém poskytuje přidání pouze MySQL databází, PostgreSQL nepodporuje.

Následuje tabulka 1 s přehledným porovnáním výše zmiňovaných systémů. Dvěma systémům chybí zpracování Cron úloh, systém Webmin tuto funkcionalitu poskytuje, ale na druhou stranu má méně uživatelsky přívětivé rozhraní. Za nejpřínosnější funkci považuji příkazovou řádku systému OpenPanel.

Název systému	Licence	Výhody	Nevýhody
VHCS	GPL	dobrá podpora, uživatelské rozhraní	administrátor nemá přístup ke službám uživatelů, chybí Cron úlohy
OpenPanel	GPL	správa plateb, příkazová řádka, PostgreSQL	chybí zálohy a Cron úlohy
Webmin	GPL	mnoho funkcí	uživatelské rozhraní
ISPConfig	GPL	možnost vytváření Cron úloh	Cron úlohy smí vytvářet pouze administrátor

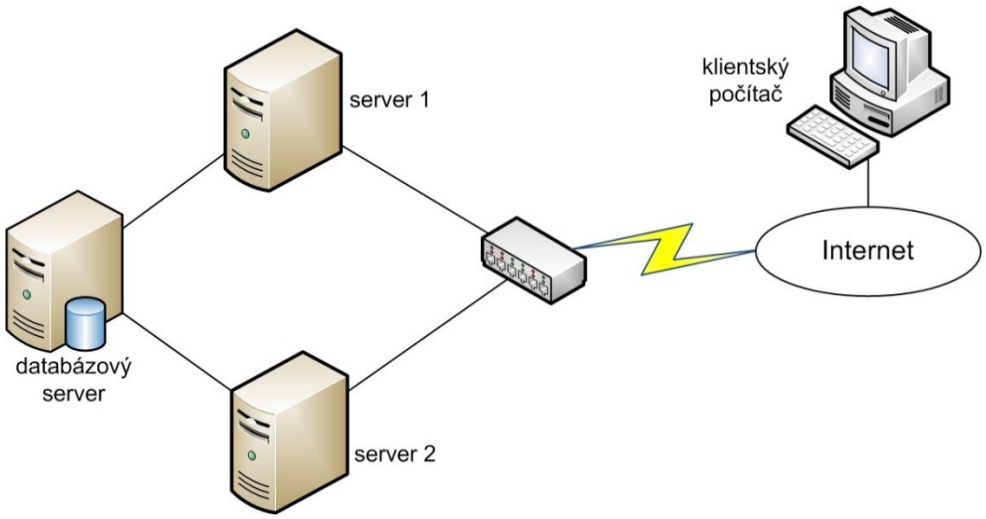
Tabulka 1: Srovnání systémů pro správu hostingu

3 Specifikace požadavků

Na systém jsou ze zadání kladeny určité požadavky. Tyto požadavky se budou následně analyzovat a implementovat do jednotlivých funkcí systému. Začnu nejdříve s uvedením základních pojmů jako je server a jeho parametry, protože je s ním celá tato práce úzce spjata. Poté se již budu věnovat samotným požadavkům na funkce systému.

3.1 Server

Serverem je obecně označován počítač, který poskytuje nějaké služby, nebo počítačový program, který dokáže tyto služby realizovat. Na Unixových systémech tyto služby zajišťuje Daemon, ve Windows Service (služba). Dá se k němu přistupovat vzdáleně ať už přes terminál, místní síť, nebo např. v případě webového serveru prostřednictvím internetu. Servery existují i jiné, než jen webové. Jsou to např. databázové, aplikační, tiskové, souborové, proxy, herní aj. Na obrázku 5 je zobrazena základní topologie vzdáleného přístupu k serveru.



Obrázek 5: Schéma propojení webového serveru s klientským počítačem

3.1.1 Webový server

Webový server je v podstatě softwarová aplikace, která čeká na určitém síťovém portu na požadavky klientů a ty pak vyhodnocuje. Klientem je většinou webový prohlížeč, ale může jím být například relace programu Telnet apod. Komerční webový server slouží většinou k pronájmu své kapacity (místa na disku) klientům, kteří si zde umístí své webové stránky. Služba, která server k tomuto účelu poskytuje, se nazývá Webhosting. Každý webový server přijímá HTTP požadavky, které vyřizuje a poté vrací odpověď počítači, který požadavek vyslal. Součástí odpovědí je i stavový kód, který určuje, zda byl požadavek v pořádku proveden. Je několik druhů stavových kódů zařazených podle druhu významu. Mezi nejdůležitější řadíme například stav 200, což značí, že byl požadavek úspěšně vykonán. Naproti tomu stavy začínající 3xx znamenají problémy s přesměrováním, stavy 4xx mohou být pro-

blémy s vyřízením požadavku, např. stav 404, který je uživatelům velmi známý z prostředí internetu, znamená, že požadovaná stránka není dostupná. A konečně mezi jedny z nejznámějších stavů jsou i začínající 5xx, což jsou interní chyby serveru.

3.1.2 Parametry serveru

Mezi důležité a sledované parametry se řadí především konektivita, která je řešena nejčastěji pomocí protokolu IPv4, pro které ale v této době docházejí rezervované adresy (kapacita pouze přibližně 4 miliardy adres), a proto se přechází na protokol IPv6. Tento nový protokol má zhruba $3{,}4\times10^{38}$ dostupných volných adres. Připojení serveru do páteřní sítě internetu je v lepším případě 1Gb/s, nebo i 100Mb/s. Co se týče technických proporcí serveru, je důležitá jeho rychlost a kapacita, která je dána výkonnými procesory, rychlými paměťmi RAM a pevnými disky, určenými k ukládání dat. Dnešní kapacita serverových pevných disků se počítá v řádech terabajtů. Příklad ukázkové hardwarové konfigurace serveru lze vidět v tabulce 2.

CPU:	2x Intel Quad-Core XEON E506, 2.13 GHz
RAM:	48 GB DDR3 ECC, zn. Kingston
HDD:	Seagate v RAID 1: 4x 750 GB, 4x 500 GB
MB:	Supermicro X8DTi, 2x CPU Socket 1366
Připojení:	1 Gbit / sek., 20 TB NIX, 500 GB zahr.
Skříň:	1U Supermicro SC113TQ-563UB, 8x Hot-swap
OS:	openSUSE 11.4 x86_64

Tabulka 2: Příklad technických parametrů Unixového serveru

3.2 Požadované funkce systému

Každý server může být jinak nakonfigurovaný, a proto může mít i jinou škálu poskytovaných služeb. Mohou být servery, které mají za úkol dělat jedinou funkci, např. databázový, který operuje pouze s databázemi a nemá nic jiného na starosti. Tyto servery (např. databázové) ale většinou hrají úlohu doplňkových služeb k serverům hlavním. Mezi hlavní služby se řadí především FTP, SVN, Cron, Databáze (nemusí být samostatný server), webový prostor, systém záloh dat, VPN, které si dále popíšeme.

3.2.1 FTP

Jedná se o protokol pro přenos souborů v rámci počítačové sítě. Využívá protokol TCP a je platformě nezávislý. Komunikace probíhá na portu TCP/21, který má za úkol řídit komunikaci a přenášet příkazy FTP a port TCP/20, sloužící k vlastnímu přenosu dat. Každému uživateli hostingu je vymezen prostor pro nahrávání a správu souborů přes FTP rozhraní, které je realizováno přes FTP klientskou aplikaci.

3.2.2 SVN

SVN je systém pro správu a verzování zdrojových kódů. Slouží jako nahrazení původního verzovacího systému CVS. Systém SVN opravuje spoustu chyb a přináší nové funkce, které starému systému chyběly. Je to multiplatformní volně dostupný systém i pro komerční využití. Hlavní funkčnost tkví v tom, že se lze vrátit zpět k jednotlivým etapám vývoje softwarového díla, protože je zaznamenávána každá změna kódu. Systém ukládá pouze jednotlivé změny kódu, nikoliv celé kódy, čímž je zajištěna úspora místa na disku serveru. Má možnost pracovat se dvěma typy úložišť, jedno je databázové, druhé souborové. S tímto systémem jsou svázány tyto pojmy:

3.2.2.1 Repozitář

Jednotlivé verze kódů se organizují do repozitáře a jsou fyzicky uloženy v souborovém systému serveru. Většinou lze říci, že každý softwarový projekt má svůj jednotlivý repozitář.

3.2.2.2 Větev

Organizuje jednotlivé repozitáře ve stromové struktuře podobné té adresářové v operačním systému.

3.2.2.3 Revize

Pokud v nějaké větvi dojde ke změně, vytvoří se nová revize v rámci celého repozitáře. Každá revize obsahuje údaje o tom, co bylo změněno, datum a čas, kdo změnu provedl. Ke každému stavu kódu se lze zpětně vrátit.

3.2.2.4 Pracovní kopie

Každý klient, který pracuje s repozitářem, má lokálně uloženu jeho pracovní kopii. Pokud klient udělá jakoukoliv změnu v této kopii, lze je promítnout i do centrálního repozitáře příkazem commit.

3.2.2.5 Commit

Jedná se o příkaz sloužící k nahrávání změn od lokálního klienta do repozitáře. Při odesílání může nastat konflikt. Konflikt nastane v případě, že odeslaná data klientem se do repozitáře nedají uložit, protože verze upravovaného souboru je jiná, než ta, kterou má před commitem klient ve své lokální kopii. Jinými slovy to znamená, že před posláním klientovy nové verze stačil dotyčný soubor dříve upravit někdo jiný.

3.2.2.6 Merge

Pomocí příkazu Merge provádíme sloučení změn repozitáře do pracovní kopie.

3.2.2.7 Changeset

Jedná se o sadu změn, které jsou prováděny v rámci celého repozitáře.

3.2.3 Cron

Pomocí Cron úloh můžeme plánovat periodické provádění serverových procesů. Jako příklad Cron úlohy může být např. periodické zasílání emailů, nebo pravidelně spouštěný skript pro údržbu databáze aj. Všechny nadefinované úlohy jsou uloženy lokálně na serveru v souboru Crontab s názvem crontab.txt. Každý řádek v tomto souboru obsahuje jednu definici úlohy, kterou lze vidět na příkladu 1.

```
A B C D E F /cesta_k_souboru
```

Příklad 1: Vzor zápisu Cron úlohy

Kde jednotlivé znaky A až F nahradíme čísly podle tabulky 3. Příklad konkrétně zadaných úloh lze vidět na příkladu 2.

Pozice	Význam	Hodnota
A	minuta	0-59
B	hodina	0-23
C	den v měsíci	1-31
D	měsíc	1-12
E	den v týdnu	0 = neděle, 1 = pondělí, 2 = úterý, 3 = středa, 4 = čtvrtek, 5 = pátek, 6 = sobota
F	rok	1970 -2099

Tabulka 3: Význam zápisu Cron úlohy

```
0 * * * * * /users/send_emails        #každou hodinu

5 2 * * * * /users/send_emails        #5 minut po druhé hodině

*/5 * * * 1 * /users/send_emails      #každých 5 minut v pondělí
```

Příklad 2: Konkrétní příklad Cron úloh

Více o tabulkách crontab lze nalézt v [4].

3.2.4 Databáze

Jednou z nejvyužívanějších služeb serverů je databáze. Je to úložiště dat, které obstarává program s názvem SŘBD. Je to rozhraní mezi aplikačními programy a uloženými daty. Databáze dělíme podle způsobu, jakým data ukládají na hierarchické, síťové, objektové, objektově relační a v naprosté většině nejpoužívanější relační databáze. Existuje mnoho komerčních SŘBD programů, jmenovitě např.

Oracle, Microsoft SQL Server, Sybase, Informix, aj. Jsou velmi drahé, ale také určené pro výkonné aplikace, které zpracovávají velké objemy databázových dat a mají mnoho funkcí navíc, které jiné SŘBD nemají. Pro nekomerční využití jsou k mání i bezplatná řešení, jako jsou např. nejrozšířenější MySQL, PostgreSQL, SQLite aj.

3.2.4.1 MySQL

MySQL databáze je volně dostupná a zcela zdarma. Je vydávána pod GPL licencí. Tvoří velkou část podílu databází na trhu, díky své lehké implementovatelnosti, snadnému nasazení a vysokému výkonu. Častým spojením technologií bývá LAMP (Linux + Apache + MySQL + PHP), ale MySQL databáze lze samozřejmě využívat i na jiných konfiguracích a systémech, než je tento příklad.

3.2.5 Webový prostor

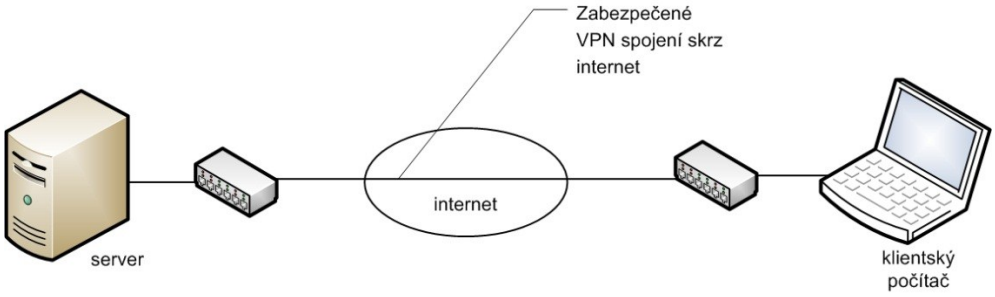
U webových serverů je kladen důraz na výkon, ale také na velmi důležitý prostor pro klientská data. Typické velikosti disků se počítají v řádech terabajtů s tím, že jednotlivým uživatelům je přiřazována kvóta, která určuje kolik prostoru můžou využívat pro své potřeby. Na kolik bude mít každý uživatel nastavenou kvótu, je už na samotné webhostingové službě.

3.2.6 Zálohování dat

Data serveru musí být určitým způsobem zabezpečena. To znamená, že musí mít ochranu proti vniknutí nepovolaných osob, které by mohly data smazat, změnit, nebo jakkoliv zneužít. Data musí být ale i pravidelně zálohována, aby byla chráněna proti nepředpokládaným výpadkům technických prvků serveru, nebo pro případ, že si uživatel sám svá data nějakým způsobem poškodí, nebo smaže a chce vše vrátit do předchozího stavu. Zálohování dat může být v případě serverů s mnoha daty velmi zdlouhavá operace, proto se provádí většinou v nočních hodinách, aby nebyl v nejvyšším provozu přes den server zatěžován. Optimální perioda zálohování je jednou denně, ale u citlivějších, často aktualizovaných dat může být nastaven i kratší interval. Aby mělo zálohování smysl, měla by být data ukládána na zvláštní fyzický disk, který by měl být fyzicky na jiném místě než primární disk, ze kterého jsou kopírována.

3.2.7 VPN

Občas je potřeba se vzdáleným serverem navázat spojení, jako bychom s ním byli spojeni ve stejné fyzické místní síti. K tomu slouží VPN. Vytvoří zabezpečené spojení, tzv. tunel mezi serverem a klientským počítačem skrz internet. Spojení je zabezpečeno šifrováním a je navázáno po ověření certifikátů obou stran. Mezi zástupci softwaru pro vytvoření VPN spojení jsou nejznámější OpenVPN, který je volně dostupný. V tomto programu lze spojení vytvořit pomocí certifikátů, sdíleného klíče, ale možné je přihlášení i se jménem a heslem. Používá port 1194 a komunikuje pomocí UDP protokolu, volitelně lze využít i TCP. Alternativou k tomuto programu může být např. Hamachi. Příklad VPN spojení je na obrázku 6.

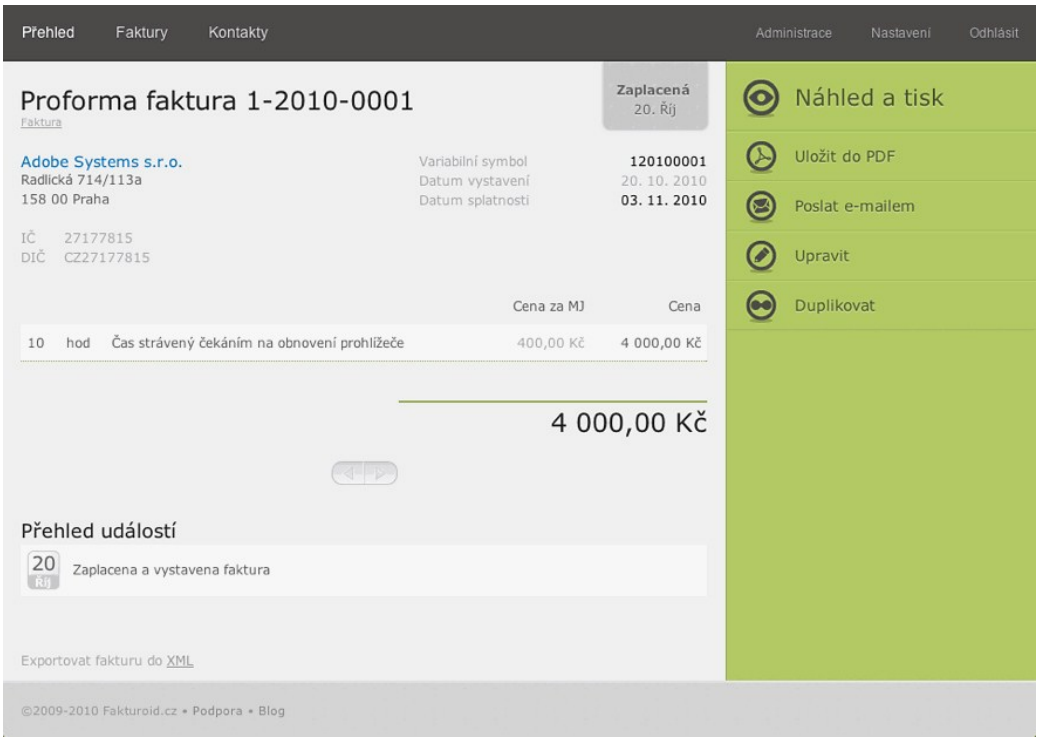


Obrázek 6: VPN spojení

3.2.8 Platby

Systém plateb za služby hostingu nebude implicitně zabudován přímo do aplikace pro správu hostingu, ale aplikace bude napojena na systém Fakturoid, ze kterého bude vzdáleně data přijímat a zobrazovat. Fakturoid je aplikace pro správu plateb a vystavování faktur, která je určena pro malé firmy a živnostníky. Aplikace je komerční a vyvinula ji skupina programátorů a je dostupná na jejich stránkách[5]. Ve Fakturoidu je možné jednoduše a přehledně spravovat platby uživatelů, např. za hostingové služby. Náhled uživatelského rozhraní aplikace je vidět na obrázku 7. Registrace je velmi jednoduchá a rychlá. Není potřeba žádné instalace, systém je dostupný na stránkách tvůrců. Má aplikace pro správu hostingu bude s Fakturoidem komunikovat pomocí architektury REST. Jakým způsobem to bude provedeno, je popsáno v pozdějších kapitolách.

[5] http://www.fakturoid.cz



Obrázek 7: Fakturoid uživatelské rozhraní

3.3 Architektura REST

Framework Ruby on Rails je založen na architektuře REST. Jak tedy tato architektura funguje a co nám přináší, bude náplní této části textu.

S příchodem Web 2.0 přišla i potřeba zlepšení komunikace mezi servery, především vzdálené volání procedur apod. Toto volání procedur je zajišťováno např. velice známým a používaným protokolem XML-RPC, nebo jeho nástupcem SOAP, které jsou orientovány procedurálně a jsou založeny na principu výměny zpráv pomocí XML přes síť, hlavně HTTP. Co se týče REST, v originálním znění Representational State Transfer, je architektura pro distribuované prostředí, která je oproti SOAP orientována datově. Distribuované prostředí je takové, kde jsou části aplikace rozděleny do několika částí, které běží na různých strojích a tyto části spolu komunikují přes síťové rozhraní. REST využíván pro snadný přístup ke zdrojům. Těmito zdroji mohou být data, nebo i stavy aplikace, které lze nějakými daty popsat. Každý zdroj má svůj jednoznačný identifikátor URI, ke kterým REST přistupuje pomocí 4 základních metod, které implementují funkce známé jako CRUD (Create, Retrieve, Update, Delete). Hlavní výhody této architektury tkví ve snadné rozšiřitelnosti psané aplikace, naopak mezi nevýhody patří skutečnost, že klient při každém požadavku musí zaslat všechny informace nutné k jeho vykonání. Více o architektuře se lze dočíst v [6].

3.3.1 Metody REST

Následuje popis jednotlivých metod, které architektura REST využívá.

3.3.1.1 GET (Retrieve)

Tato základní metoda slouží k získávání dat konkrétního zdroje. Setkáváme se s ní téměř každý den, kdykoli si prohlížíme webové stránky, neboť se jedná o klasické vyžádání stránky směrem k serveru. Jednoduchým příkladem je zaslání požadavku GET na adresu nějakého webového serveru, tak jako je ukázáno na příkladu 3.

```
GET http://www.sampleserver.com/page.html HTTP/1.0
Referer: http://www.sampleserver.com/home.html
```

Příklad 3: Požadavek GET

Zde je vidět požadavek na server sampleserver.com na stránku page.html pomocí protokolu HTTP ve verzi 1.0. Položka Referer určuje, ze kterého zdroje byl požadavek vyslán, což může být například užitečné při vypisování chybových hlášení apod. Pro HTTP protokol ve verzi 1.1 může požadavek vypadat jako na příkladu 4. Více o HTTP protokolu se lze dočíst v [7] a [8].

```
GET /page.html HTTP/1.1
Host: www.sampleserver.com
```

Příklad 4: Požadavek GET pro HTTP 1.1

Server vrací při úspěšném požadavku stavový kód 200, což znamená zprávu OK. Za těmito dvěma příkazy musí být vložen prázdný řádek.

3.3.1.2 POST (Create)

Stejně jako metoda GET, je i tato metoda uživateli, resp. webovému vývojáři známá, minimálně aspoň z prostředí formulářů, ve kterých touto metodou data odesíláme směrem k serveru. Při volání této metody není znám přesný identifikátor zdroje, protože zdroj ještě v té chvíli neexistuje. Proto je volen jako společný identifikátor "endpoint". Po odeslání by měl server vrátit stavový kód 201 v němž lze odeslat identifikátor nově vytvořeného zdroje. Takovýto požadavek lze vidět na příkladu 5. Content-Type nám značí MIME typ dat z formuláře a Content-Length určuje délku zasílaných dat. Mezi těmito příkazy a konkrétními daty musí být vložen rovněž prázdný řádek.

```
POST /add.php HTTP/1.1
Host: www.sampleserver.com
Content-Length: 29
Content-Type: application/x-www-form-urlencoded

pole1=hodnota1&pole2=hodnota2
```

Příklad 5: Požadavek POST

3.3.1.3 DELETE (Delete)

Pomocí volání tohoto HTTP požadavku zdroj smažeme. Dále můžeme vidět na příkladu 6 ukázku takovéhoto požadavku.

```
DELETE /user/10 HTTP/1.1
Host: www.sampleserver.com
```

Příklad 6: Požadavek DELETE

3.3.1.4 PUT (Update)

Metoda PUT je velice podobná metodě POST pro vytvoření nových dat. Jediný rozdíl je v tom, že v požadavku zašleme i identifikátor zdroje, který chceme změnit. Metoda PUT je vidět na příkladu 7.

```
GET /edit.php HTTP/1.1
Host: www.sampleserver.com
Content-Length: 29
Content-Type: application/x-www-form-urlencoded

pole1=hodnota1&pole2=hodnota2
```

Příklad 7: Požadavek PUT

3.4 XML-RPC

XML-RPC je komunikační protokol, který slouží ke vzdálenému volání procedur. Touto procedurou může být např. funkce aplikace třetí strany, nebo aplikace, které potřebujeme zaslat data z jiné aplikace, přičemž jejich implementace může být odlišná, dokonce napsána i v jiných programovacích jazycích. V tom je velká síla tohoto protokolu, je multiplatformní. Komunikace probíhá, jak je již z názvu patrné, zasíláním zpráv ve struktuře XML. Ukázka XML zprávy je vidět na příkladu 8 níže.

```
POST /sample HTTP/1.0
User-Agent: Mozilla/5.0 (Windows; U; ...)
Host: sub.sampleserver.com
Content-Type: text/xml
Content-length: 291

<?xml version="1.0"?>
<methodCall>
  <methodName>SampleClass.sampleMethod</methodName>
    <params>
      <param>
        <value><string>lorem ipsum...</string></value>
      </param>
    </params>
```

```
</methodCall>
```

Příklad 8: XML zpráva - požadavek

Zpráva obsahuje HTTP hlavičku s adresou serveru pro příjem požadavku. Dále je obsahem již struktura XML zprávy. První řádek tvoří specifikace, že jde o XML verze 1.0, další řádek již uvozuje párový tag <methodCall>, do kterého je vnořen párový tag <methodName>, v němž je název volané třídy a její metody. Tag <params> uvozuje skupinu jednotlivých parametrů, které jsou metodě předávány. Počet těchto parametrů není limitován. Po zpracování požadavku serverem, je přijata odpověď, jejíž tělo je vidět ve výpisu níže. Odpověď serveru se od požadavku klienta liší uvozovacím párovým tagem <methodResponse>, ve kterém jsou předány parametry odpovědi směrem ke klientovi. Ukázka XML odpovědi je v příkladu 9.

```
<?xml version="1.0"?>
<methodResponse>
       <params>
          <param>
             <value><boolean>true</boolean></value>
          </param>
       </params>
</methodResponse>
```

Příklad 9: XML zpráva - odpověď

3.5 Apache

Apache je multiplatformní HTTP server, který má v dnešní době majoritní podíl využití u webových serverů. Vyvíjí ho firma The Apache Software Foundation a hlavní výhodou je jeho otevřenost, je poskytován zcela zdarma pod licencí GPL. Plně dodržuje standard HTTP/1.1 (RFC 2616). Podporuje velký počet webových sídel na jednom stroji (Virtual Host). Dokáže spolupracovat s mnoha skriptovacími jazyky jako je PHP, PERL, Python apod. V současnosti se Apache nachází ve verzi 2.2.17. Pro nastavení serveru existují různé distribuční a grafické nástroje, nebo můžeme upravit rovnou konfigurační soubory s názvy apache2.conf (httpd.conf), ports.conf a conf.d/charset.

3.5.1 Moduly

Apache server je modulární, čili lze si stáhnout pouze základní balík a další moduly doinstalovat podle potřeb. Tyto moduly vytváří komunita po celém světě, jedná se tedy o softwarové balíky třetích stran. Mezi hlavní moduly patří tyto:

3.5.1.1 Moduly jádra

Je to modul, který je načten ze dvou modulů core.c, httpd_core.c a musí být vždy připojen k jádru Apache staticky. Je pro chod serveru Apache nezbytný. Funkce tohoto modulu jsou dostupné ve všech serverech Apache.

3.5.1.2 Standardní moduly

Tyto moduly jsou součástí standardní verze Apache a jsou vyvíjeny The Apache Software Foundation. Narozdíl od modulu jádra, můžou být jakékoliv standardní moduly odstraněny správcem serveru, aniž by byl narušen jeho chod. Nejčastěji tomu tak dochází kvůli snížení rozsahu využité paměti pro každou instanci serveru.

3.5.1.3 Moduly třetích stran

Jedná se o moduly, které nevytvořila skupina Apache Group, ale programátoři, kteří se o Apache zajímají a rozšiřují tímto způsobem jeho funkcionalitu. Nejsou obsaženy v distribuci Apache a musí být získány odděleně. Některé populární moduly se za léta vývoje staly standardními a jsou již v těchto distribucích obsaženy.

3.5.2 Nastavení Apache

Nastavení serveru probíhá, jak již bylo zmíněno, v souboru apache2.conf, resp. httpd.conf. Po stáhnutí distribuce je server již přednastaven a většinou stačí případně jen změnit pár nastavení, nebo server rovnou používat. V tabulce 4 jsou uvedeny nejběžnější parametry a jejich popis. Více o serveru Apache lze dohledat v [3] a [9].

Direktiva	Popis
KeepAlive *On*	Perzistentní spojení (HTTP/1.1)
HostnameLookups *Off*	Překlad IP adres na jména
LogLevel *warn*	Logování činnosti serveru
ServerTokens *Full*	Hlavička informací serveru
ServerName *www.sample.com*	Pojmenování serveru
ServerAdmin *admin@sample.com*	Kontakt při chybách serveru
ErrorDocument 404 */error.html*	Chybová hlášení

Tabulka 4: Parametry nastavení Apache

3.6 Uživatelé a jejich práva

Je kladen požadavek na rozdělení funkčnosti systému mezi typy uživatelů, resp. typy uživatelských práv. V systému jsou pouze dva typy uživatelů. Je to role “uživatel” a “administrátor”. Každá z nich má definovány akce, které smí vykonávat a jsou v uživatelském rozhraní přístupné. Dále tyto akce pro obě role popíši.

3.6.1 Akce pro roli uživatele

Uživatel bude moct provádět základní akce dostupné v sekci Základní a Nastavení. Sekce Administrace mu přístupná není. Tyto sekce jsou popsány v kapitole návrhu implementace. V sekci Základní a Nastavení je možno přidávat, upravovat a mazat záznamy pouze ty, které "vlastní" sám uživatel, resp. nemůže provádět žádné akce na datech jiných uživatelů. V seznamu níže jsou vypsány všechny akce, které uživatel může s daty provádět.

- přihlášení a odhlášení
- vytvoření FTP uživatele
- editace FTP uživatele
- smazání FTP uživatele
- vytvoření MySQL uživatele
- smazání MySQL uživatele
- editace MySQL uživatele
- vytvoření MySQL databáze
- smazání MySQL databáze
- vytvoření domény
- editace domény
- smazání domény
- vytvoření subdomény
- editace subdomény
- smazání subdomény
- vytvoření doménového aliasu
- editace doménového aliasu
- smazání doménového aliasu
- smazat zálohu
- tvorba Cron úlohy
- smazání Cron úlohy
- editace Cron úlohy
- tvorba SVN repozitáře
- editace SVN repozitáře
- smazání SVN serveru
- tvorba SVN uživatele
- editace SVN uživatele
- smazání SVN uživatele

3.6.2 Akce pro roli administrátora

Administrátorovi je již sekce Administrace přístupná. Zde má přístup ke všem záznamům všech uživatelů, které může upravovat, mazat, ale také přidávat. Výpis všech funkcí dostupných administrátorovi jsou v následujícím seznamu.

- přihlášení a odhlášení

- vytvoření uživatele
- mazání uživatele
- změna údajů uživatele
- vytvoření FTP uživatele
- editace FTP uživatele
- smazání FTP uživatele
- vytvoření MySQL uživatele
- smazání MySQL uživatele
- editace MySQL uživatele
- vytvoření MySQL databáze
- smazání MySQL databáze
- aktivace statistik pro doménu
- vytvoření domény
- editace domény
- smazání domény
- vytvoření subdomény
- editace subdomény
- smazání subdomény
- vytvoření doménového aliasu
- editace doménového aliasu
- smazání doménového aliasu
- smazat zálohu
- tvorba Cron úlohy
- smazání Cron úlohy
- editace Cron úlohy
- tvorba SVN repozitáře
- editace SVN repozitáře
- smazání SVN repozitáře
- tvorba SVN uživatele
- editace SVN uživatele
- smazání SVN uživatele
- tvorba novinky
- editace novinky
- smazání novinky
- zaslání emailu, hromadného emailu
- zobrazení logu
- smazání logu

4 Analýza požadavků

V předchozí kapitole byly popsány všechny funkce, které jsou po systému pro správu hostingu požadovány. Hlavním úkolem je nyní tyto požadavky analyzovat a vytvořit tak přípravu pro další etapu vývoje informačního systému, kterou je návrh. Více o etapách tvorby informačních systémů lze najít v [1].

4.1 Lineární zápis typů entit:

Lineární zápis typů entit je jeden z prvků úplného konceptuálního datového modelu. Obsahem jsou jednotlivé entity a jejich atributy. Pokud se jedná o primární klíč, který je zároveň identifikátorem jednoznačnosti entity, je označen tučně. Cizí klíč je odlišen podbarvením. Kompletní datový slovník a další součásti analýzy, jako jsou minispecifikace, podrobnější DFD diagramy a stavové diagramy, jsou zpracovány v příloze 1.

Primární klíč, cizí klíč

Users (**id**, customer_id, username, password, admin, enabled, last_ip, note, email, created_at, updated_at)

Failed_logins (**id**, username, password, ip_address, created_at, updated_at)

Crons (**id**, user_id, name, hours, minutes, days, months, dotw, log, path, enabled, created_at, updated_at)

Account_infos (**id**, user_id, used_space, used_db_space, used_svn_space, files_count, current_login_attemps, log, created_at, updated_at)

Account_options (**id**, user_id, max_ftp_users, max_databases, max_db_users, max_crons, max_svn_repositories, max_login_attemps, created_at, updated_at)

Logs (**id**, user_id, message, created_at, updated_at)

Options (**id**, value, key, created_at, updated_at)

Messages (**id**, user_id, priority, message, sent, created_at, updated_at)

Domains (**id**, user_id, name, statistics, ssl, safe_mode, created_at, updated_at)

Subdomains (**id**, user_id, name, domain_id, ssl, stats, safe_mode, created_at, updated_at)

Domain_aliases (**id**, user_id, name, domain_id, created_at, updated_at)

Ftp_users (**id**, user_id, username, password, uid, gid, dir, ul_bandwidth, dl_bandwidth, status, created_at, updated_at)

Web_backups (**id**, user_id, size, created_at, updated_at)

Mysql_backups (**id**, user_id, size, created_at, updated_at)

Mysql_databases (**id**, user_id, name, created_at, updated_at)

Mysql_users (**id**, user_id, username, created_at, updated_at)

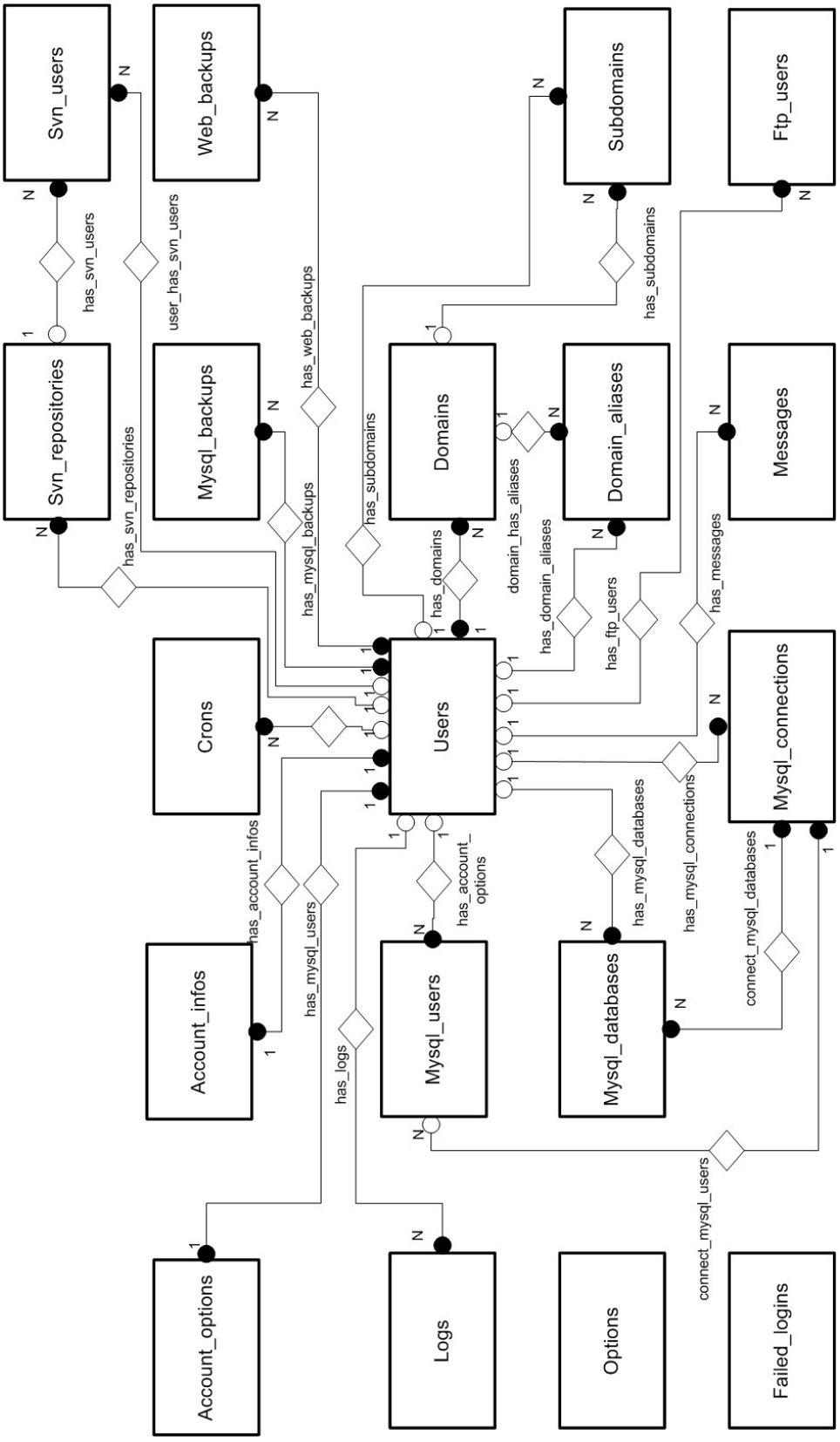
Mysql_connections (**id**, user_id, mysql_user_id, mysql_database_id, created_at, updated_at)

Svn_repositories (**id**, user_id, name, size, created_at, updated_at)

Svn_users (**id**, svn_repository_id, user_id, name, created_at, updated_at)

4.2 ER diagram:

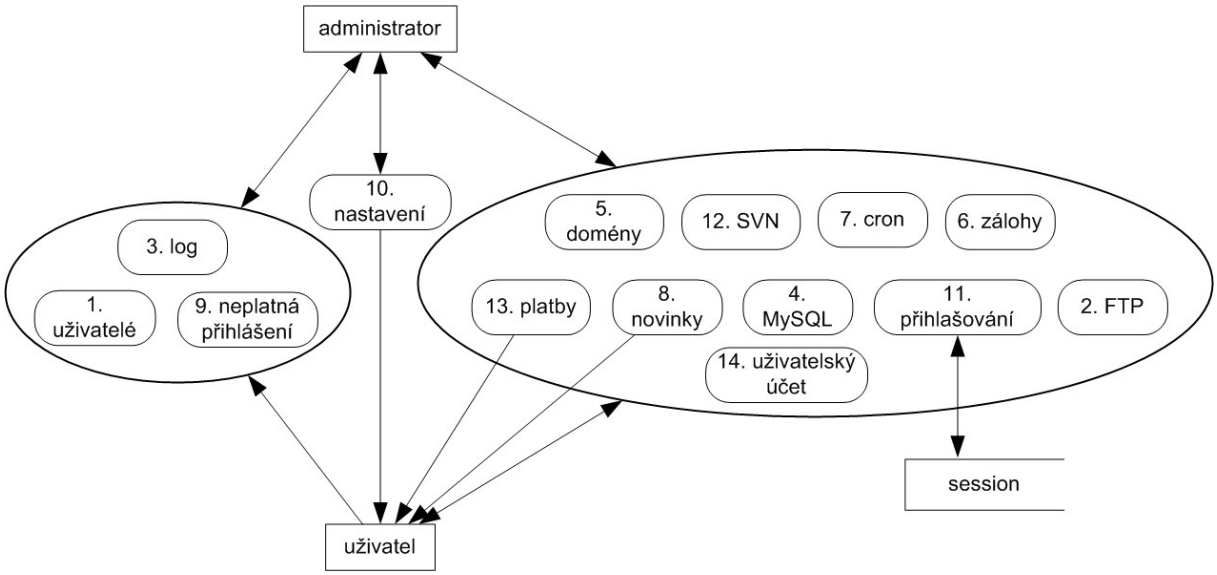
ER diagram je grafický náhled vztahů entit, zachycuje datový model. Zobrazuje i kardinalitu a povinnost jednotlivých entit. Diagram lze vidět na obrázku 8.



Obrázek 8: ER diagram

4.3 DFD

DFD je diagram datových toků, který znázorňuje externí entity, které přicházejí do styku s procesy. V aplikaci pro správu hostingu jsou těmito externími entitami administrátor a uživatel. Tyto entity mají přístup k celkem 14 procesům, které jsou dále rozděleny na podprocesy. Základní diagram 0. úrovně je na obrázku 9. Znázornění detailních DFD 1. úrovně lze dohledat v příloze 1.



Obrázek 9: DFD diagram 0. úrovně

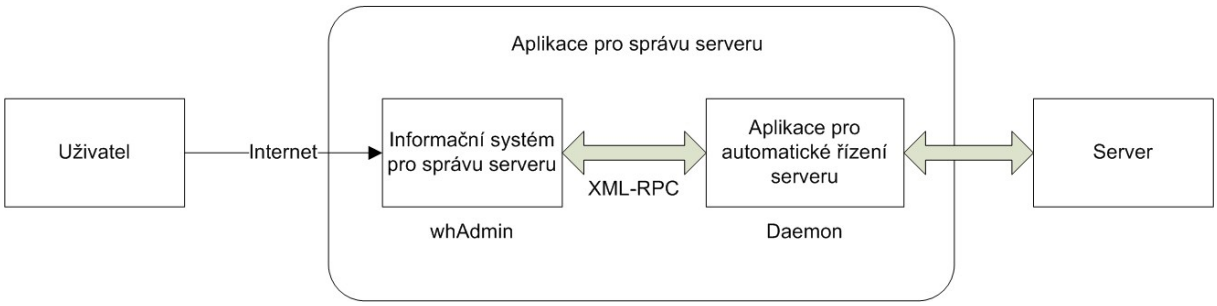
Jak je na první pohled patrné, z externí entity administrátor vedou oboustranné datové toky ke všem procesům. To znamená, že administrátor do těchto procesů vstupuje a mění jejich stavy a data a zároveň z nich i data přijímá. Uživatel je, co se týče datových toků, v určitých procesech omezen.

5 Návrh implementace

Další etapou vývoje informačního systému je návrh implementace, založená na předchozí analýze a návrhu. V jednotlivých podkapitolách nejprve nastíním na jaké části bude aplikace rozdělena a jakým způsobem spolu budou komunikovat, dále volbu implementačního jazyka a použité architektury MVC a adresářovou strukturu systému. V neposlední řadě bude také navrhnuto uživatelské rozhraní systému s popisem jednotlivých částí.

5.1 Rozvržení aplikace

Aplikace bude rozdělena na dvě části, jak je patrné z obrázku 10. Uživatel se pomocí internetu připojí do informačního systému, ve kterém provede požadované operace a nastavení serveru. Poté informační systém zašle požadavek na provedení operací aplikaci pro automatické řízení serveru. Tento požadavek bude zasílán pomocí metody XML-RPC, která byla popsána v kapitole dříve. Aplikace dále provede požadované operace a nastavení přímo se serverem.



Obrázek 10: Rozvržení aplikace

5.2 Struktura informačního systému

Informační systém, se kterým přijde do styku koncový uživatel, bude založen na architektuře MVC a napsán bude v jazyce Ruby za použití frameworku Ruby on Rails. Adresářová struktura pro tento framework je v NetBeans IDE následující.

- Controllers – obsahuje kontroléry, které jsou součástí MVC architektury
- Helpers – helper je třída, která nám přidává nějakou funkčnost, kterou využíváme na více místech aplikace. Tato složka obsahuje všechny helpery.
- Models – modely, které jsou také součástí MVC architektury
- Views – obsahuje soubory pohledů, jež tvoří prezentační část MVC architektury
- app/mailers – knihovny pro zasílání emailů
- Configuration – konfigurační soubory pro nastavení aplikace, jazykovou lokalizaci, ale obsahuje i důležitý soubor pro definici směrování v aplikaci (routes.rb)
- Database Migrations – soubory migrací, jež definují databázové schéma
- Libs – dodatečné knihovny

- Logs – veškeré logování činnosti serveru
- Public – jediná složka, která je přístupná uživateli. Obsahuje soubory kaskádových stylů, javascripty a obrázky
- Test Files – obsahuje testovací sady aplikace
- Scripts – jejím obsahem je script rails, který je důležitý pro chod aplikace
- Documentation – adresář určený pro dokumentační soubory aplikace
- Vendor – obsahem jsou dodatečné zásuvné moduly aplikace
- Libraries – součástí je základní jádro programovacího jazyka Ruby ve verzi 1.8.7

5.2.1 Návrh uživatelského rozhraní

Do systému se uživatelé budou přihlašovat přes přihlašovací formulář, který je vidět na obrázku 11. Aplikace byla pojmenována whAdmin (webhosting administration) a náhled na její rozhraní, které je po přihlášení aplikace dostupné, je na obrázku 12. Rozhraní bude dostupné v českém, anglickém a německém jazyce, které se budou moci plynule přepínat.

5.2.1.1 Prvky rozhraní

Aplikace je rozdělena na tři části. Vrchní část, tzv. hlavička (header) obsahuje logo, v pravé části informace o uživateli a volba odhlášení z aplikace. Dále hlavička obsahuje pás s hlavním nabídkou, které je popsáno později. Druhá část se skládá z vpravo umístěné nabídky rychlého menu, kde jsou nejčastější volby pro rychlou práci s rozhraním a hlavní část, která slouží jako hlavní zobrazovací pole pro právě zvolenou sekci. Pod rychlým menu se nachází panel s plynule rotujícími novinkami.

Posledním prvkem rozhraní je dolní pás s menším logem a copyrightem aplikace. Všechny tyto části jsou uživateli po přihlášení vždy zobrazeny.

5.2.1.2 Nabídka

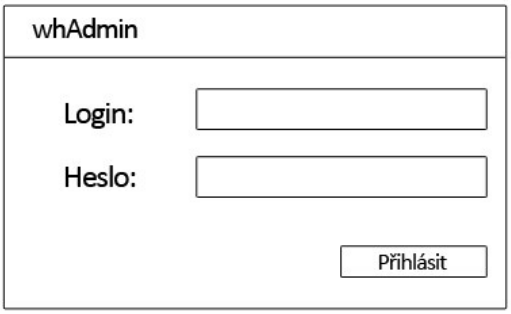
Nabídka je rozdělena na tři sekce a to Základní, Nastavení a Administrace. Administrace bude přístupná pouze pro uživatele s právy administrátora.

V části Základní uživatel uvidí novinky, což jsou informace ohledně činnosti hostingu. Dále zde najde své informace o účtu uživatele, funkci pro zaslání dotazu na podporu hostingu. Najde zde odkazy na službu FTP a phpMyAdmin, což je aplikace pro správu MySQL databází. Poslední volbou je možnost zobrazení informací o nainstalované verzi PHP.

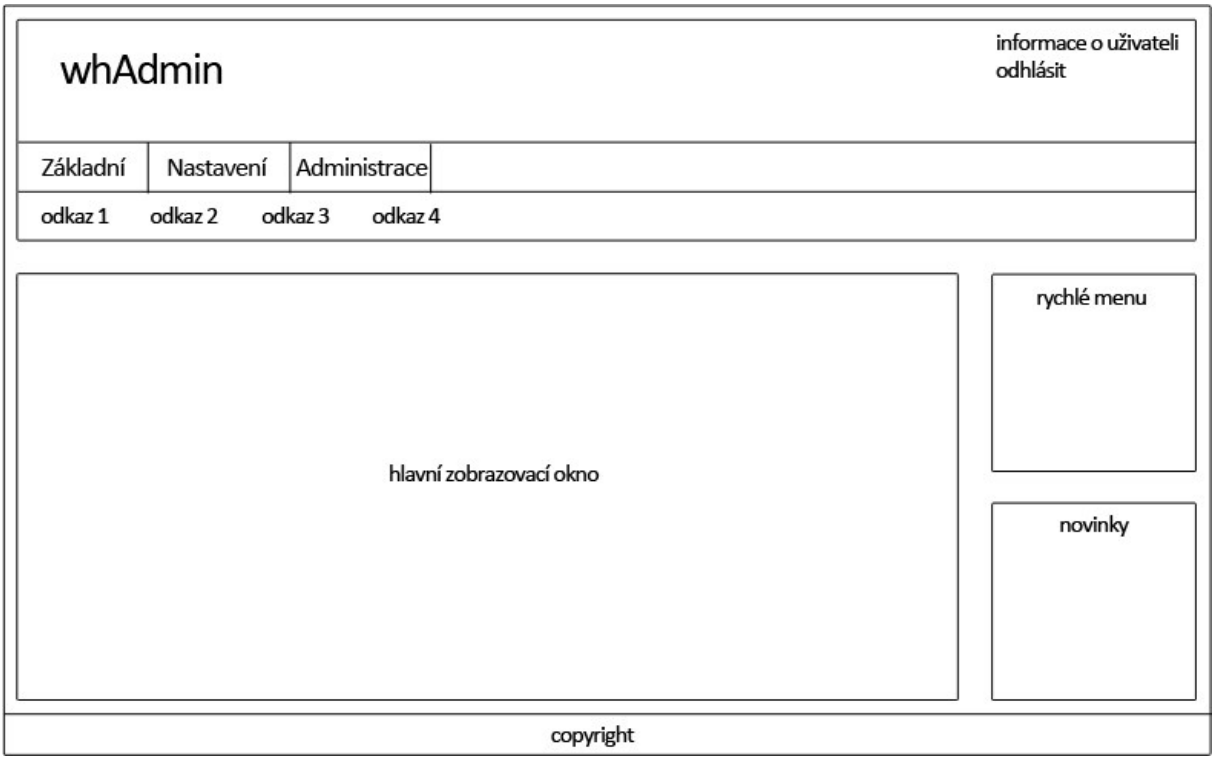
Část Nastavení je určena k samotnému nastavování parametrů služeb serveru. Obsahuje nastavení FTP uživatelů, jejich přidávání, editaci a mazání, správu MySQL uživatelů a jejich databází, správu domén, subdomén a doménových aliasů. Dále si zde uživatel může přidávat Cron úlohy, může je i editovat a samozřejmě mazat, spravovat SVN repozitáře a jejich uživatele. Má přehled o zálohách svého webového prostoru a MySQL databází, má možnost tyto zálohy mazat. Poslední dvě volby se věnují zobrazení plateb za služby hostingu a nastavení osobního účtu, kde si uživatel může změnit své heslo.

V sekci Administrace má administrátor přístup ke správě všech uživatelů. Tyto uživatele zde může vytvářet, upravovat a mazat stávající. Dále zde může spravovat domény, subdomény a doménové alia-

sy, FTP účty, může spravovat novinky serveru, Cron úlohy všech uživatelů, zálohy, SVN repozitáře a jejich uživatele, MySQL uživatele a jejich databáze. Administrátor má přehled o platbách všech uživatelů za hostingové služby, může nahlédnout do logu činnosti serveru a upravovat nastavení. Jako doplňkovou administrační funkci může využít zaslání emailů svým zákazníkům.



Obrázek 11: Přihlašovací formulář



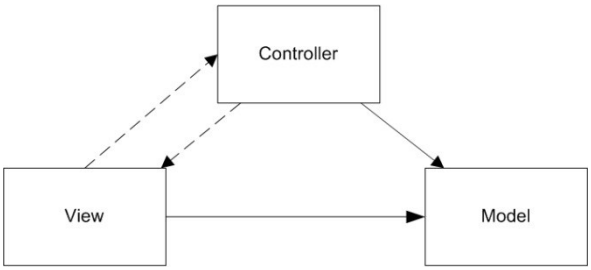
Obrázek 12: Uživatelské rozhraní aplikace

5.3 Třívrstvá architektura MVC

MVC je třívrstvá architektura, skládající se z Modelu, View a Controlleru a někdy se mu říká také Model 2. Hlavní podstatou je oddělení prezentační logiky aplikace od aplikační. K tomu slouží rozdělení aplikace do tří vrstev, které dále popíši. Jak je patrné z obrázku 13, Controller má přímý přístup

k Modelu a může díky němu upravovat informace objektů a View má přímý odkaz na Model, aby mohl zobrazovat jeho data. Neexistuje přímá vazba Modelu na ostatní dvě komponenty. Typická akce mezi vrstvami může probíhat takto:

1. Uživatel vyvolá akci ve View v uživatelském rozhraní.
2. Controller zachytí tuto akci.
3. Dále se v Controlleru rozhodne, zda se upraví objekt v Modelu, nebo se ihned vrátí výsledek do View.
4. Uživateli se provedené změny a informce o nich vypíší na obrazovku ve vrstvě View.



Obrázek 13: Architektura MVC

5.3.1 Model

Touto vrstvou je tzv. doménový model, který modeluje vztahy reálného světa. Nejen že reprezentuje informace s níž aplikace pracuje (data z databází) pomocí objektů, ale obsahuje např. i validační pravidla pro tvorbu a editaci objektů, business pravidla, která určují za jakých podmínek se objekt může měnit a vytvářet.

5.3.2 View

View slouží jako prezentační vrstva, která na výstupu poskytuje informace uživateli o objektech, např. ve formě zobrazení výsledných dat pomocí HTML stránky, nebo XML apod. Zobrazuje informace z Modelu a další prvky, které z něho nemusí nutně pocházet.

5.3.3 Controller

Controller má na starosti tok událostí a aplikační logiku. Propojuje prezentační část (View) s datovou strukturou (Model) a stará se tak o celkovou funkčnost modelu. Zpracovává data z Modelu a zasílá vrstvě View výsledné informace, která je poté zobrazí.

5.4 Implementační jazyk

Implementační jazyk části aplikace pro automatické řízení serveru byl zvolen Ruby ve verzi 1.8.7 a pro druhou část informačního systému framework Ruby on Rails ve verzi 3.0.7.

Ruby je plně objektově orientovaný interpretovaný skriptovací programovací jazyk. V současné době začíná být velice populární, a to nejen díky na něm postaveném frameworku Ruby on Rails pro tvorbu

webových aplikací, ale také hlavně pro svou sílu ulehčovat práci programátorům. Tato přednost spočívá v jeho syntaxi, která dovoluje psát kód velmi agilně a některá řešení problémů jsou mnohem kratší a sofistikovanější, než tomu bývá u konkurenčních jazyků. Díky své interpretovanosti odpadá nutnost jazyk kompilovat, čili změny ve zdrojovém kódu lze ihned vidět. Ruby podporuje dynamické datové typy, regulární výrazy, ale také psát webové aplikace díky frameworku Ruby on Rails. Ruby on Rails je plnohodnotný MVC framework určený pro vývoj internetových aplikací. Využívá REST architekturu, která byla popsána výše. Více o jazyce Ruby se lze dočíst ve [2] a o Ruby on Rails v [5].

5.5 Instalace Ruby

Pro instalaci Ruby v operačním systému Windows existuje automatizovaný instalátor, který je k dostání na internetu[6]. V operačních systémech na bázi Linuxu je pro instalaci platný následující příkaz.

```
sudo apt-get install ruby-full
```

5.5.1 Gemy

Ruby on Rails není framework, který by měl všechny funkce již obsaženy. Každá funkcionalita navíc se musí nainstalovat pomocí tzv. gemů, což jsou balíčky které nám je zpřístupní. Existuje velká knihovna těchto gemů a je volně šiřitelná buďto přes web[7], nebo se dají instalovat přes příkazovou řádku v operačním systému. Ruby on Rails obsahuje některé gemy již v základním gemu rails. Následuje stručný výčet těchto základních gemů.

- Action Pack
  - Action Controller
  - Action Dispatch
  - Action View
- Action Mailer
- Active Model
- Active Record
- Active Resource
- Active Support
- Railties

Hlavní Ruby on Rails gem se instaluje pomocí následujícího příkazu.

```
gem install rails
```

[6] http://www.rubyinstaller.org
[7] http://www.rubygems.org

Tento příkaz nainstaluje spolu i výše vypsané základní balíčky funkcí. Jako doplňkové gemy byly dále doinstalovány jquery-rails, který nahradí implicitní AJAX knihovnu Prototype a sqlite3-ruby, což zpřístupní SQLite databázi pro vývojové prostředí.

5.6 Vývojové prostředí

Jako vývojové prostředí pro psaní kódu aplikace bylo využito NetBeans IDE ve verzi 6.9.1. Byla použita distribuce přímo pro Ruby on Rails. V tomto prostředí je řada nástrojů, která s tímto jazykem úzce spolupracuje. Obsahuje rozhraní pro spouštění příkazů jazyka, samozřejmostí je i podpora "našeptávání". NetBeans IDE bylo zvoleno z důvodu poskytnutí tohoto prostředí zcela zdarma na internetu[8] a pro svou jednoduchou použitelnost a vhodnost pro mnou vytvářenou aplikaci. Toto prostředí obsahuje i lokální server WEBrick[9], na kterém byla zkoušena webová část aplikace při jejím vývoji.

[8] http://www.netbeans.org
[9] http://microjet.ath.cx/WebWiki/WEBrick.html

6 Popis implementace

V této kapitole se budu věnovat již samotnému řešení aplikace a její implementaci. Nejdříve popíši aplikaci pro automatické řízení serveru, jak k ní informační systém přistupuje a jakým způsobem volá její funkce. Poté provedu rozbor jednotlivých funkcí systému jak z pohledu části s informačním systémem, ke které přistupují uživatelé, tak z pohledu aplikace pro automatické řízení serveru.

6.1 Daemon

Daemonem se nazývá aplikace, která obsluhuje uživatelovy požadavky. Tyto požadavky jsou volány vzdáleně, uživatel nemá přímý přístup k Daemonu. V našem případě je Daemonem aplikace pro automatické řízení serveru.

6.1.1 XML-RPC

Jak již bylo napsáno v předchozí kapitole, volání Daemona bude realizováno přes XML-RPC architekturu. Konkrétně se jedná o řešení klient – server, kde na straně klienta stojí informační systém, který zasílá požadavky k serveru, který je zpracovává. Na straně serveru je Daemon. Ukázka kódu konkrétní implementace klienta je na příkladu 10.

```
def daemon action, *args
    begin
      server = XMLRPC::Client.new(get_o('host'),"/",
get_o('port_daemon'))
    rescue
      log t(:xmlrpc_not_connect, :exc => "#{$!}")
      flash[:error] = t(:daemon_error)
      return false
    end

    begin
      status = server.call(action, *args)
      return status
    rescue
      log t(:server_call_error, :exc => "#{$!}")
      flash[:error] = t(:daemon_error)
      return false
    end
end
```

Příklad 10: Klientská metoda pro zaslání XML-RPC požadavku

```
# Vytvoreni XML-RPC serveru
rpc = XMLRPC::Server.new($config['xmlrpc'][0]['port'][0].to_i)

# Pridani trid pro volani funkci
rpc.add_handler('users', $users)
rpc.add_handler('domains', $domains)
rpc.add_handler('mysql', $mysql)
rpc.add_handler('crons', $crons)
rpc.add_handler('backups', $backups)
rpc.add_handler('svn', $svn)
rpc.serve
```

Příklad 11: Serverová metoda pro přijímání XML-RPC požadavku

Na příkladu 11 je vidět zdrojový kód pro vytvoření XML-RPC serveru. Jsou zde zaregistrovány třídy pro obsluhu funkcí serveru. Nakonec je zapnuto naslouchání serveru, resp. server je připraven přijímat požadavky.

6.1.2 Struktura

Všechny Ruby zdrojové kódy jsou organizovány do adresářové struktury, která je popsána níže.

- apache_templates – obsahuje šablony s konfiguračními soubory Apache serveru
- doc – zde je umístěna programátorská dokumentace
- models – obsahuje modely
- web_template – obsahem je šablona, která se automaticky překopíruje do složky s novou doménou, po jejím vytvoření. Jedná se o soubor index.html, který říká, že si uživatel zatím nevytvořil žádnou stránku.

Hlavní adresář aplikace obsahuje tyto Ruby soubory:

- main.rb – probíhá zde spouštění XML-RPC serveru, spouštění zálohování a přepočítávání informací o uživatelích.
- config.rb – konfigurační soubor, napojení na databázi a modely.
- utils.rb – obsahuje pomocné funkce Daemona

Dále hlavní adresář obsahuje již soubory s výkonným kódem aplikace, jako jsou domains.rb, backups.rb apod.

6.1.3 Volání procedury

Při každé akci, která je provedena v informačním systému whAdmin, je zároveň zavolána pomocí XML-RPC metoda Daemona, která provede příslušnou operaci na serveru. Tyto vzdálené procedury jsou volány z kontrolérů systému whAdmin. Ukázka volání Daemona je na příkladu 12.

```
daemon("users.init_new_user", @user.id)
```

Příklad 12: Volání metody Daemona (tvorba nového uživatele)

6.2 Funkce aplikace

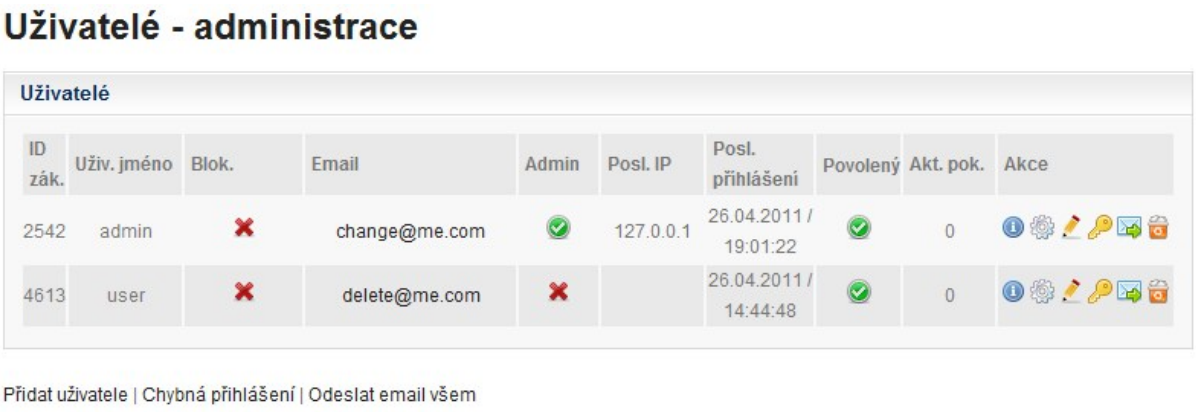
V této kapitole popíši všechny funkce aplikace, jakým způsobem jsou implementovány a jaké aktivity provádějí se samotným serverem. Vždy bude ke každé akci nejprve popsána část informačního systému a poté i část aplikace pro automatické řízení serveru. Vložené ukázky formulářů jednotlivých funkcí budou vždy společné pro přidání i editaci záznamu s tím rozdílem, že při editaci budou ve formuláři již data předvyplněna a připravena k úpravě a některá data nebudou editovatelná, resp. nebudou ani zobrazena vstupní pole pro úpravu. Budou zde také vloženy ukázky uživatelského rozhraní, ovšem kvůli velkému rozsahu nebudou všechna. Kompletní návod k systému je v uživatelské příručce a v programátorské dokumentaci.

6.2.1 Správa uživatelů

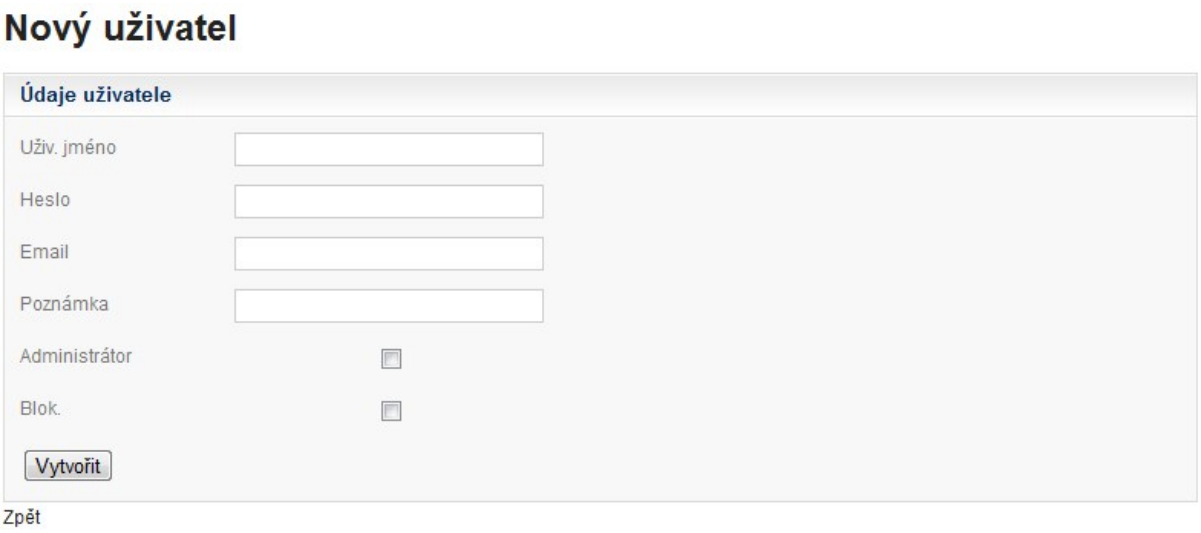
Základní funkcí systému je správa uživatelů. Každý uživatel je zároveň i zákazníkem hostingu, využívá a platí jeho služby. V systému nebude řešena volná registrace, přidávat uživatele může jen administrátor.

6.2.1.1 Informační systém

Správa uživatelů je dostupná v Administraci pouze uživateli v roli administrátor. Administrátorovi se zobrazí tabulka s výpisem všech uživatelů a jejich informací, jak je vidět na obrázku 14.

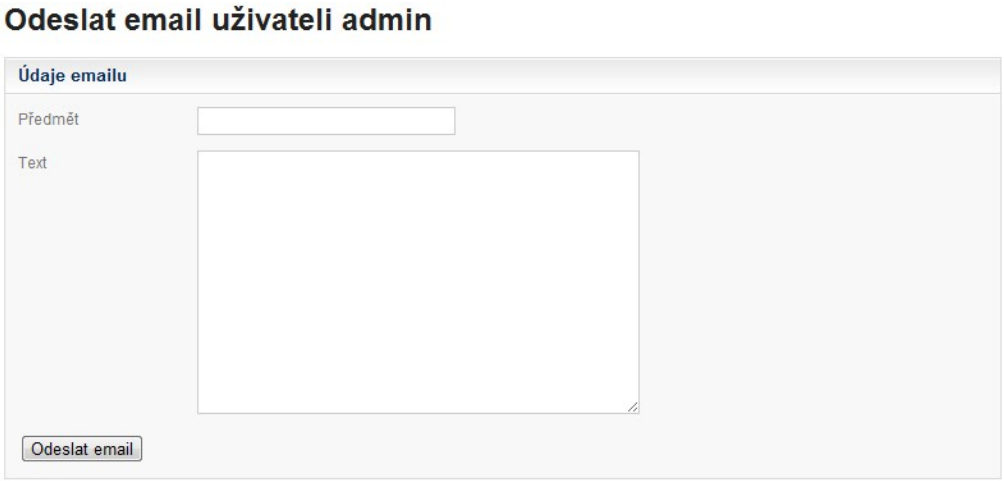


Obrázek 14: Tabulka uživatelů



Obrázek 15: Formulář uživatele

Při přidání nebo úpravě uživatele se zobrazí formulář, který je vidět na obrázku 15. Administrátor vyplní údaje a odešle. Poté se mu zobrazí výpis nového nebo upraveného záznamu a dále je provedeno přesměrování na výpis všech záznamů uživatelů. U každého uživatele je volba odeslat email, pod níž se skrývá formulář na obrázku 16. Po vyplnění údajů a odeslání se administrátorovi zobrazí hlášení a proběhne přesměrování na seznam uživatelů. Za symbolem klíče, který je u každého uživatele se skrývá funkce pro vyresetování a následnému zaslání nového hesla na uživatelův email a symbol ozubeného kola slouží pro nastavení voleb a omezení pro uživatele, jako je např. maximální počet SVN repozitářů, maximální počet neplatných pokusů o přihlášení apod.



Obrázek 16: Formulář poslání emailu uživateli

6.2.1.2 Aplikace pro automatické řízení serveru

Při vytvoření uživatele je z kontroléru pro uživatele zavolána metoda Daemona `init_new_user` s parametrem uživatelova id. Daemon poté pomocí modelu User najde požadované jméno uživatele

a vytvoří v adresáři /srv/www na serveru adresář. Při mazání je akce obdobná s tím rozdílem, že je zavolána metoda `delete_user` a pomocí ní Daemon adresář smaže. Proběhne také smazání všech uživatelových Cron úloh, databází, databázových uživatelů, domén a subdomén apod. Poté se restartuje Apache server. Metodu pro vytváření uživatele lze vidět níže na příkladu 13.

```
def init_new_user user_id
    user = User.find user_id
    dir = WhUtils.get_u_dir user

    # Vytvoreni hlavniho adresare s weby
    WhUtils.create_dir dir

    # Vytvoreni systemovych adresaru
    WhUtils.create_dir WhUtils.get_logs_dir user
    WhUtils.create_dir WhUtils.get_stats_dir user
    WhUtils.create_dir WhUtils.get_backups_dir user

    true
end
```

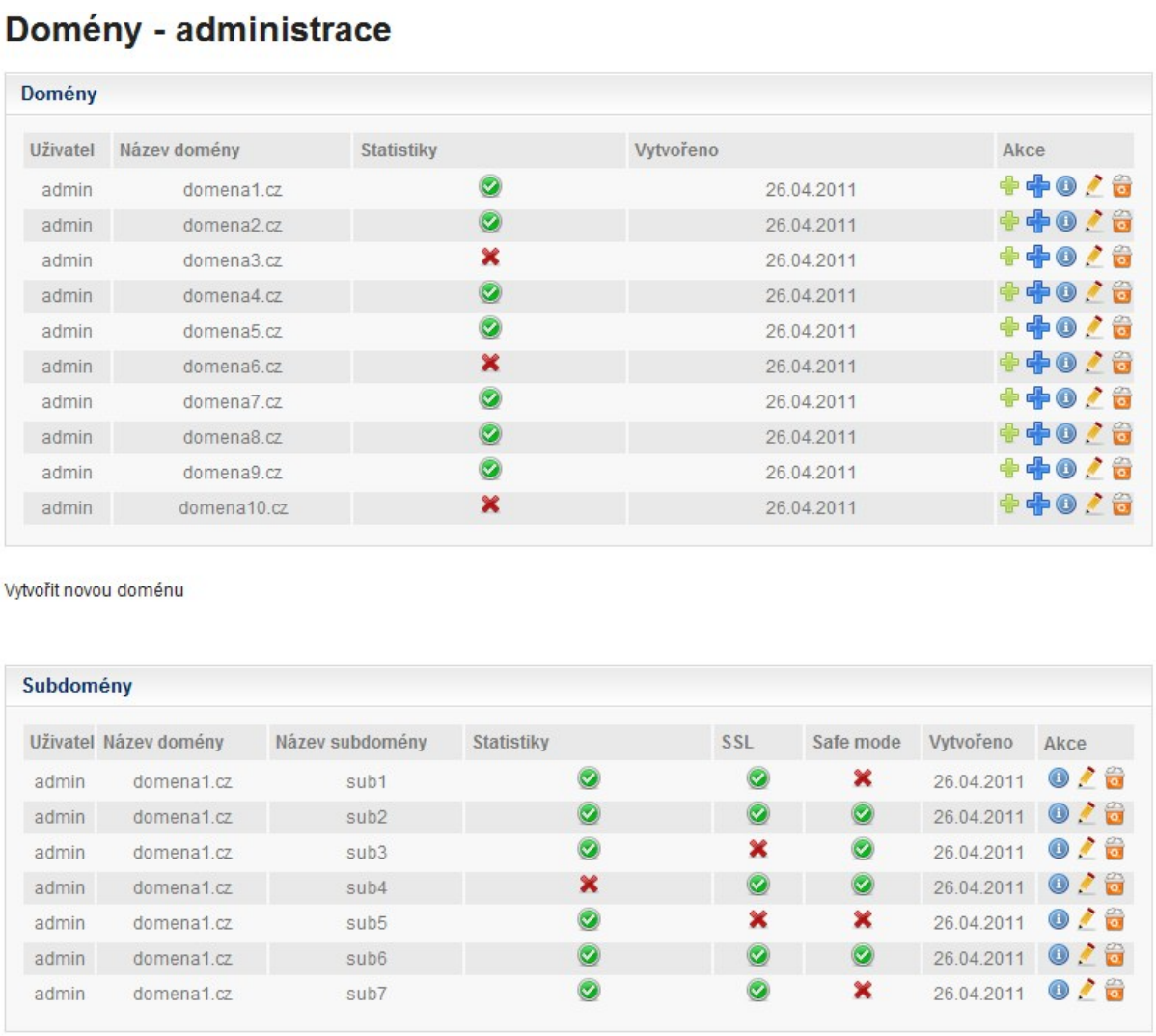
Příklad 13: Vytvoření nového uživatele

6.2.2 Domény, subdomény a doménové aliasy

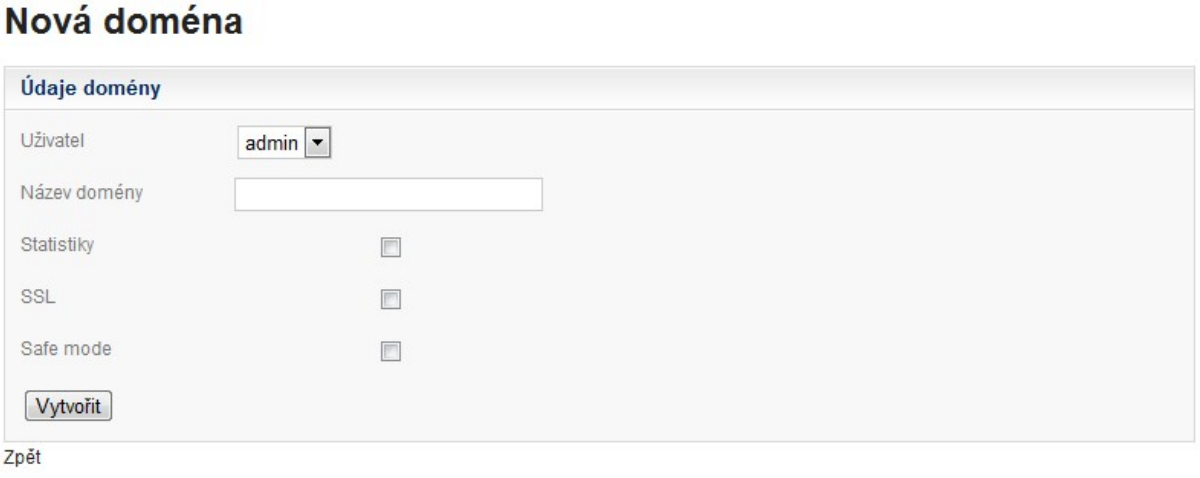
Každý uživatel si bude moct přidávat nové domény, subdomény a doménové aliasy a spravovat již vytvořené.

6.2.2.1 Informační systém

Správa domén je pro uživatele dostupná v Nastavení ve volbě Domény SSL a statistiky a pro administrátory v sekci Administrace, volba Domény. Jak je vidět na obrázku 17, na stránce se v informačním systému zobrazí jednotlivé domény od jednotlivých uživatelů a pod touto tabulkou další dva seznamy, které obsahují jednotlivé subdomény a doménové aliasy (na obrázku 17 není) všech uživatelů. Obrázek ukazuje výpisy pro administrátora. V uživatelské sekci není zobrazen sloupec “Uživatel” a jsou zobrazeny jen záznamy platné pro aktuálně přihlášeného uživatele. Ikonka “zelený křížek” umožňuje přidání subdomény pro vybranou doménu a “modrý křížek” je určen pro přídání doménového aliasu. Formulář pro přidání nové domény je na obrázku 18 a pro přidání subdomény na obrázku 19.

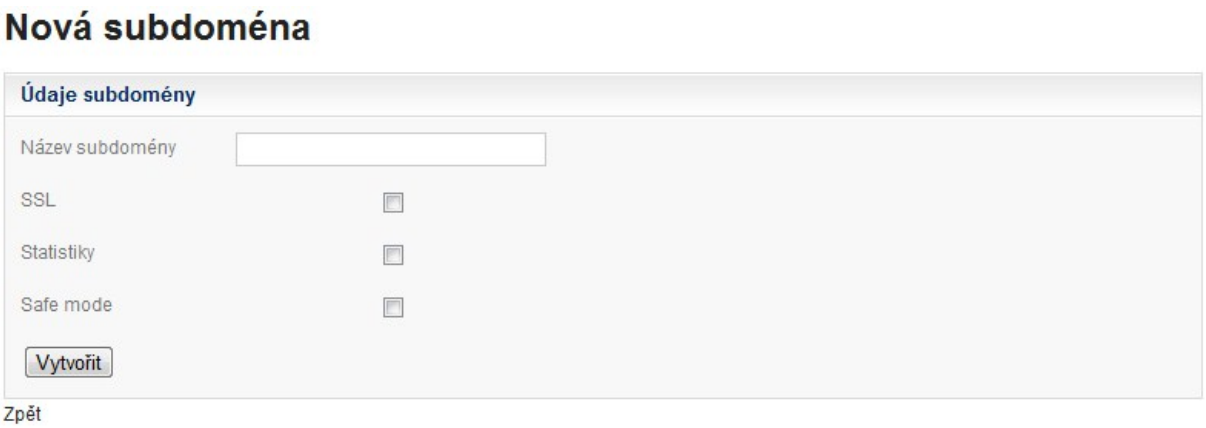


Obrázek 17: Tabulka domény



Obrázek 18: Formulář nová doména

Ve formuláři pro novou doménu administrátor vybere, pro kterého uživatele ji vytvoří. Tato volba je přístupná pouze administrátorovi a zákazníkům (uživatelům) není ve formuláři zobrazena, záznam se jim přidruží automaticky. Poté může volitelně zvolit zda se budou pro doménu generovat statistiky, SSL protokol a Safe mód. Při tvorbě nové subdomény je formulář stejný pro obě role. Stačí jen vyplnit název subdomény a nastavit zda bude povolen SSL protokol, statistiky a Safe mód. Po odeslání obou formulářů se zobrazí kontrolní výpis údajů a poté proběhne přesměrování na výpis domén.



Obrázek 19: Formulář nová subdoména

Formulář pro doménový alias a akce pro jeho vytvoření jsou obdobné jako u subdomény, s tím rozdílem, že po kliknutí na "modrý křížek" do formuláře uživatel zadá pouze jméno doménového aliasu. Upravovat nelze názvy domén a subdomén, lze pouze nastavit parametry SSL, statistiky a Safe mode (tato volba je dostupná pouze pro administrátora jak v editaci, tak při vytváření domény a subdomény).

6.2.2.2 Aplikace pro automatické řízení serveru

Při vytvoření domény v informačním serveru je v kontroléru pro domény zavolána metoda Daemona `new_domain` s parametrem id domény. Server XML-RPC požadavek zachytí a metoda pro vytvoření nové domény vytvoří adresář na serveru pro doménu, logy, statistiky. Poté je do adresáře zkopírována webová šablona a jsou nastavena práva adresáře. Je vytvořen konfigurační soubor serveru Apache a poté je restartován. Ukázku kódu Daemona pro vytvoření domény je na příkladu 14.

```
def new_domain domain_id
    domain = Domain.find domain_id
    unless domain
      return false
    end

    dir = WhUtils.get_dom_www_dir domain
```

```
    WhUtils.create_dir dir
    WhUtils.create_dir WhUtils.get_dom_logs_dir domain
    WhUtils.create_dir WhUtils.get_dom_stats_dir domain

    FileUtils.cp_r  Dir.glob($config['dirs'][0]['web_template'][0]  +
'/*'), dir
    WhUtils.set_dir_perm(dir)
    create_domain_vhost domain
    WhUtils.restart_apache
    true
end
```

Příklad 14: Tvorba nové domény

Vytváření subdomény je obdobné jako u domény. Vytváření doménového aliasu probíhá tak, že je zavolána metoda `update_domain_aliases`, která aktualizuje konfigurační soubor Apache pro danou doménu a zapíše do ní nové doménové aliasy. Po smazání domény jsou smazány všechny její subdomény a doménové aliasy. Dojde také ke smazání veškerých dat, které byly v těchto adresářích obsaženy. Pro úpravu domén a subdomén slouží metody `update_domain_ssl` apod., které pouze upraví některé parametry, které byly popsány výše.

6.2.3 FTP účty

Vytvořením FTP účtu dojde k zpřístupnění FTP služby. K přístupu do ní slouží uživatelům aplikace třetí strany, nazvaná net2ftp[10], dostupná na internetu. Odkaz na tuto aplikaci je umístěn v menu Základní.

6.2.3.1 Informační systém

Na obrázku 20 je tabulka vytvořených FTP uživatelů. Jsou zde vidět i atributy jako jsou UID, GID, UL bandwidth a DL bandwidth. Tyto atributy jsou však při vytváření nového FTP uživatele přístupné pouze administrátorovi. Pokud si FTP tvoří obyčejný uživatel, který nemá administrátorská práva, dosadí do těchto atributů implicitní hodnoty, které lze nastavit v sekci Nastavení. Pro každý atribut tam existuje samotný záznam, který lze změnit.

[10] http://www.net2ftp.com

Obrázek 20: Tabulka FTP uživatelé

6.2.3.2 Aplikace pro automatické řízení serveru

Aplikace Daemona není v tomto případě volána, protože Pure-FTPd[11] server je nastaven tak, aby data pro přihlášení uživatelů (jména, hesla apod.) čerpal z databázové tabulky, která je společná i pro informační systém whAdmin. Resp. když uživatel vytvoří FTP uživatele, je možné se okamžitě přihlásit pomocí FTP klienta (např. net2ftp[12]).

6.2.4 SVN

Uživatelé si mohou vytvořit pouze omezený počet SVN repozitářů. Tato volba může být u každého uživatele odlišná a nastavit ji lze v administraci uživatelů. Počet uživatelů jednotlivých repozitářů je již neomezený. Toto omezení je opět z důvodu kapacity disku serveru. Je na každém administrátorovi, aby tento počet zvolil podle možností, jaké server má. Pokud je hodnota nastavena na 0, je množství SVN repozitářů neomezené.

6.2.4.1 Informační systém

Ukázku tabulek s repozitáři a uživateli lze vidět na obrázku 21. U každého repozitáře je ikonka "zeleného křížku", která umožňuje přidat neomezené množství uživatelů a propojit je s repozitářem. Pokud bude smazán repozitář, budou odstraněni i všichni SVN uživatelé, kteří jsou k němu přidruženi.

[11] http://www.pureftpd.org/project/pure-ftpd
[12] http://www.net2ftp.com

Obrázek 21: Tabulka SVN

6.2.4.2 Aplikace pro automatické řízení serveru

Po vytvoření repozitáře je zavolána metoda `new_repository`, která je v příkladu 15. Repozitáře se ukládají na serveru do adresáře /srv/svn/. Tato metoda vytvoří adresář s repozitářem. Dalšími metodami jsou `delete_repository` pro smazání repozitáře a `rename_repository` pro přejmenování repozitáře.

```
def new_repository repository_id
    repo = SvnRepository.find repository_id
    return false unless repo
    dir = WhUtils.get_svn_repo_dir repo
    delete_repository repository_id if File.directory? dir
    WhUtils.create_svn_repo dir
    true
end
```

Příklad 15: Vytvoření SVN repozitáře

Metoda pro vytvoření SVN uživatele k danému repozitáři je v příkladu 16. Je volána při vytváření, editaci i mazání SVN uživatelů.

```
def update_svn_access repository_id
    repo = SvnRepository.find repository_id
    return false unless repo

    users = ''
    repo.svn_users.each do |svn_u|
      auth = "#{svn_u.name} = #{svn_u.password}"
      users = users + "#{auth}\n"
    end
```

```
    FileUtils.cp_r Dir.glob(WhUtils.get_svn_templates_dir + '*'),
WhUtils.get_svn_repo_conf_dir(repo)

    passwd = File.open(WhUtils.get_svn_passwd_file_path(repo),
'r').read
    WhUtils.translate passwd, {:users => users}

    pf = File.open WhUtils.get_svn_passwd_file_path(repo), 'w'
    pf.write passwd
    pf.close

    true
end
```

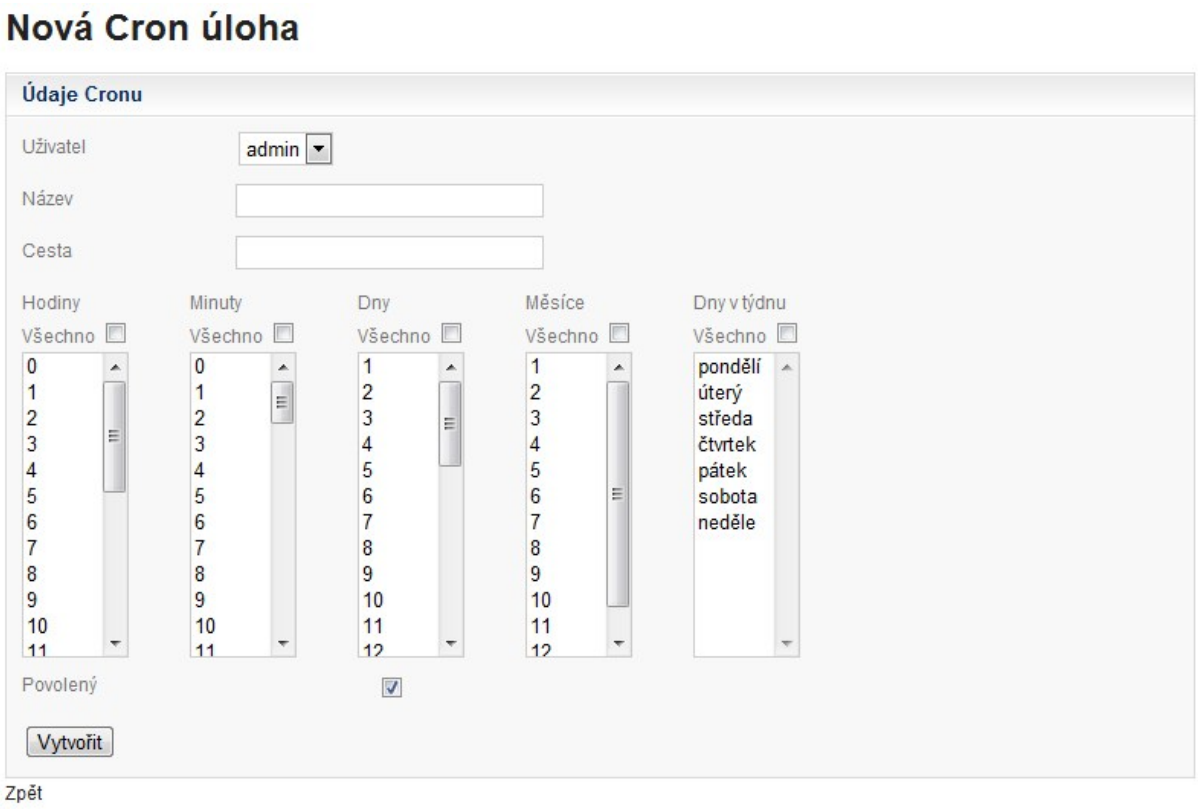
Příklad 16: Aktualizace SVN uživatelů

6.2.5 Cron úlohy

Tvorbu a správu Cron úloh považuji za důležitou součást aplikace pro vzdálené řízení serveru, i když je některé ze srovnávaných populárních systémů neobsahovaly. Každý uživatel má možnost vytvářet úlohy pouze v omezeném počtu. Toto množství může mít každý uživatel nastaveno jiné, záleží na konkrétním administrátorovi, jaké zvolí.

6.2.5.1 Informační systém

Na obrázku 22 je formulář pro vytvoření Cron úlohy. Pro administrátora je navíc zobrazena volba, pro kterého z uživatelů Cron úlohu vytvoří. Dále je nutno zadat cestu k volanému souboru, název úlohy. Prostřední blok se týká samotného nastavení období, kdy se úloha bude spouštět a periodicky opakovat. Jsou to parametry pro hodinu, minutu, den, měsíc a den v týdnu. Možnost má uživatel vybrat i více voleb z těchto seznamů, nebo rovnou zaškrtnout, že chce všechny. Systém nakonec tyto parametry naformátuje a zašle aplikaci pro automatické řízení serveru. Poslední volba je pro určení, zda má být úloha ihned povolena, nebo ji uživatel povolí až následně.



Obrázek 22: Formulář Cron úlohy

6.2.5.2 Aplikace pro automatické řízení serveru

Na příkladu 17 je ukázka metody Daemona pro tvorbu nové Cron úlohy. Zavolána je z kontroléru pro Cron úlohy. Metoda je společná pro tvorbu i editaci. Pokud je Cron editován, bude opět zavolána tato metoda s tím, že pokud již daný Cron v systému existuje, bude smazán a nahrazen novým s novými parametry. Tvorba Cron úlohy probíhá tak, že se nejdříve podle parametrů vytvoří definice Cron úlohy a poté se zapíše do souboru, který je umístěn v /etc/cron.d/nazev_cronu.id.

```
def new_cron cron_id, enabled
    cron = Cron.find cron_id
    return false unless cron

    File.delete(WhUtils.get_cron_path(cron)) if File.exists?
WhUtils.get_cron_path(cron)

    enabled ? e = '' : e ='#'
    cron_str = "#{e}#{cron.minutes} #{cron.hours} #{cron.days}
#{cron.months} #{cron.dotw} #{$config['apache'][0]['user'][0]} php
#{cron.path}"

    File.open WhUtils.get_cron_path(cron), "w" do |cfile|
```

```
      cfile.puts cron_str
    end
    true
  end
```

Příklad 17: Přidání Cron úlohy

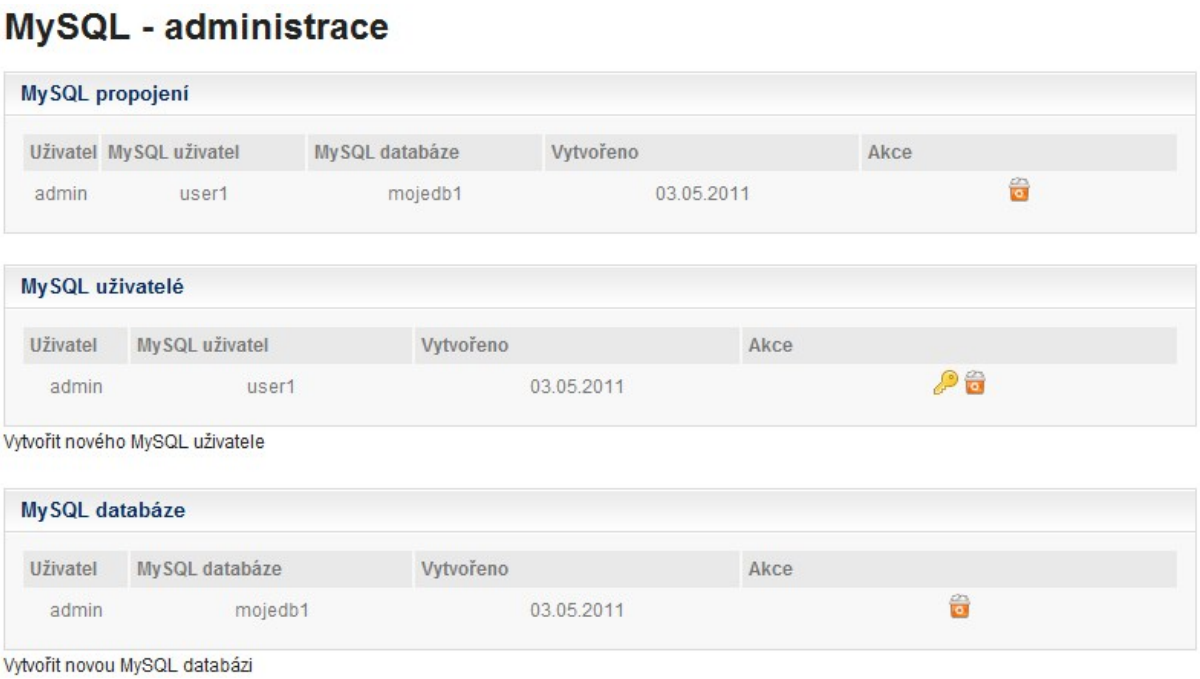
Mazání úloh je možno buďto samostatně, k tomu slouží metoda `delete_cron` a nebo hromadně po smazání uživatele pomocí `delete_all_crons`.

6.2.6 MySQL uživatelé a databáze

Každý uživatel má práva pro vytvoření MySQL databáze, administrátor tyto databáze může vytvářet pro všechny uživatele. Každá databáze je určena právě jednomu MySQL uživateli, kterého je nutné nejdříve vytvořit. Tomuto uživateli se poté databáze přiřadí. Jednotliví MySQL uživatelé mohou mít více databází.

6.2.6.1 Informační systém

Tabulky pro tvorbu MySQL uživatelů a databází lze vidět na obrázku 23. První tabulka obsahuje záznamy o propojení databází s uživateli. Pokud uživatel smaže některého z MySQL uživatelů, jeho databáze z bezpečnostních důvodů zůstane nesmazána, musí ji smazat sám dodatečně. Pokud uživatel bude chtít vytvořit novou databázi, musí mít nějakého existujícího MySQL uživatele, kterému databázi přiřadí. Editovat lze pouze heslo MySQL uživatele. Tato volba se skrývá pod ikonou „zlatého klíče“.



Obrázek 23: Tabulka MySQL

6.2.6.2 Aplikace pro automatické řízení serveru

Po zavolání metody Daemona s názvem `new_user` je proveden SQL příkaz CREATE USER s parametry jména a nešifrovaného hesla. Ukázka metody je na příkladu 18.

```
def new_user mysql_user_id, clear_passsword
    user = MysqlUser.find mysql_user_id
    return false unless user

    MySQLUser.connection.execute("CREATE USER '#{user.username}'@'%'
IDENTIFIED BY '#{clear_passsword}';")
    true
end
```

Příklad 18: Vytvoření MySQL uživatele

V příkladu 19 je ukázka metody pro tvorbu nové MySQL databáze. Provede se databázový SQL příkaz `CREATE DATABASE`. V případě smazání databáze se provádí příkaz `DROP DATABASE`. S databází se zároveň vytvoří i propojení s MySQL uživatelem pomocí příkazu `GRANT ALL ON` v metodě `assign_db_to_user`. Zrušení propojení uživatele s databází se provede příkazem `REVOKE ALL ON` v metodě `unassign_user_from_db`.

```
def new_database mysql_database_id
   database = MysqlDatabase.find mysql_database_id
   return false unless database

   MySQLUser.connection.execute("CREATE DATABASE #{database.name};")
   true
end
```

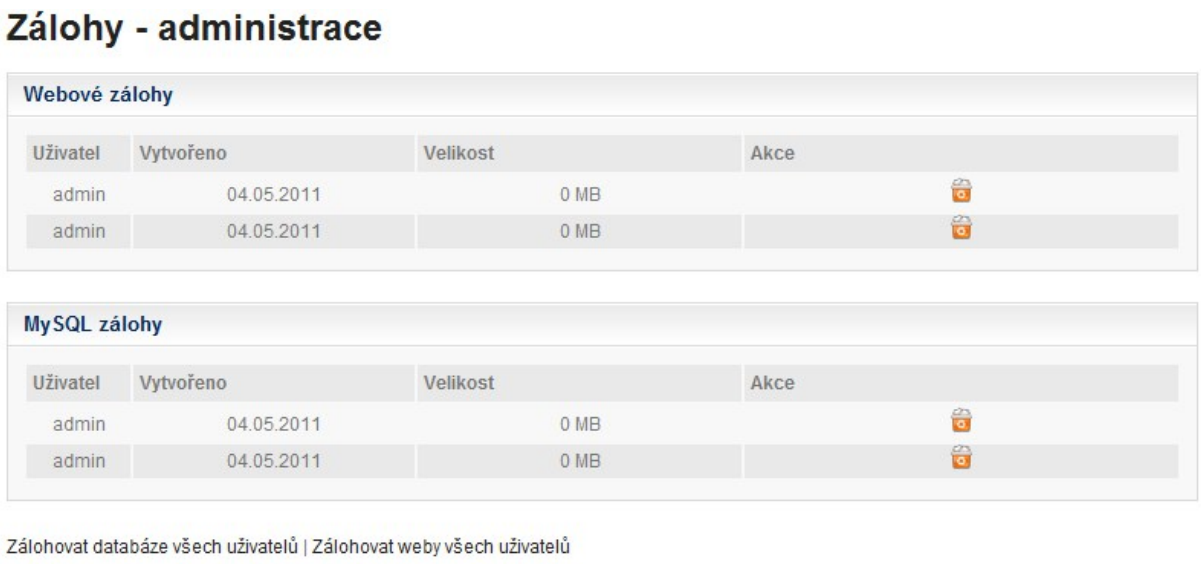
Příklad 19: Vytvoření nové databáze

6.2.7 Zálohy

Na serveru jsou prováděny automatické zálohy dat, ať už to jsou webové stránky, nebo databáze. Tyto zálohy jsou prováděny periodicky a každý z uživatelů jich může mít pouze omezené množství (z důvodu kapacity úložného prostoru). Pokud se jejich počet rovná limitu, je automaticky odstraněna nejstarší záloha. Tímto způsobem si uživatel může obnovit svá data podle času, jenž si vybere.

6.2.7.1 Informační systém

Obrázek 24 ukazuje jednotlivé zálohy webového prostoru a MySQL databáze seřazené podle data, ve kterém byly vytvořeny. S každou z nich lze provést celkem tři akce. Zálohu je možné rozbalit, nahradit, kopírovat, nebo smazat.



Obrázek 24: Tabulka zálohy

6.2.7.2 Aplikace pro automatické řízení serveru

Při smazání zálohy je záznam smazán z databáze systému a rovněž je záloha smazána i fyzicky z disku serveru. Toto smazání provede daemon v metodě `delete_web_backup` pro webové zálohy a `delete_db_backup` pro MySQL zálohy. Metoda mazání webových záloh je na příkladu 20.

```
def delete_web_backup backup_id
    backup = WebBackup.find backup_id
    return false unless backup

    dir = WhUtils.get_web_backup_dir backup
    FileUtils.rm_r dir if File.directory? dir
    true
end
```

Příklad 20: Smazání webové zálohy

Zálohování webových prostorů uživatelů může být v případě objemnějších dat časově náročné, proto jsou spouštěny např. dvakrát denně pomocí Cronu, který zavolá soubor Daemona `main.rb --backup` a provede tak zálohy. Administrátor má však možnost provést okamžitou zálohu webů i databází. Slouží k tomu hypertextové odkazy pod tabulkami s výpisem záloh. Při aktivování této funkce dojde ke spuštění jedné z funkcí `backup_all_users_web` nebo `backup_all_users_db`.

6.2.8 Platby

Systém plateb nebude impementován implicitně v informačním systému, ale bude napojen na externí aplikaci Fakturoid. Systém bude získávat data plateb uživatelů z Fakturoidu pomocí architektury

REST a bude je zobrazovat v uživatelském rozhraní. Uživatel tak bude má přehled o svých platbách na jednom místě a nebude se muset přihlašovat přímo do Fakturoidu, do kterého nebude mít přístup.

6.2.8.1 Informační systém

Uživatel si bude moct výpis těchto plateb zobrazit v Nastavení ve volbě Platby. Zobrazí se mu tabulka s platbami, která bude načtena z aplikace Fakturoid. Tabulka s výpisem je vidět na obrázku 25. Na každou fakturu je možné nahlédnout, nebo ji stáhnout ve formátu PDF.

Obrázek 25: Platby

6.2.8.2 Aplikace pro automatické řízení serveru

Aplikace pro automatické řízení serveru není pro tuto funkci volána.

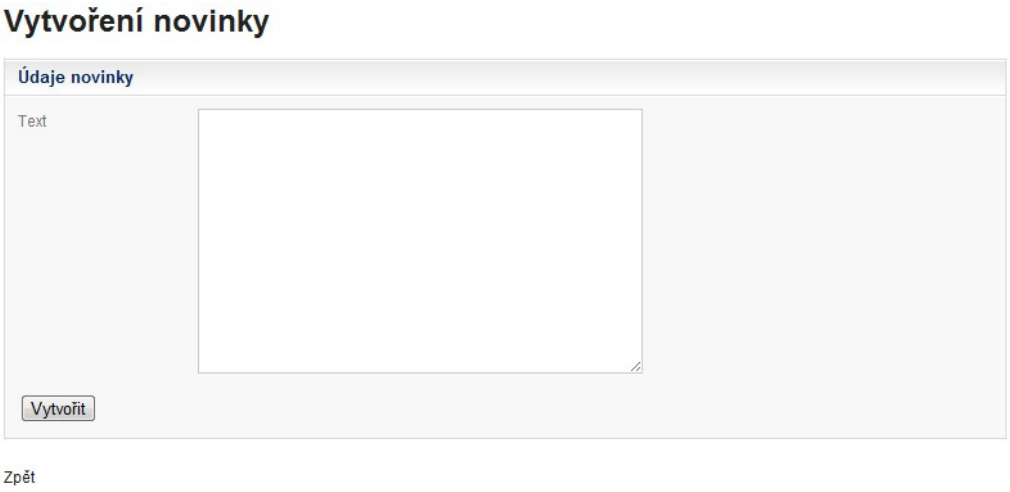
6.2.9 Novinky

Novinky jsou v informačním systému určeny k informování zákazníka (uživatele) o změnách serveru, případných plánovaných výpadcích apod. Jejich výpis je zobrazován hned v úvodu po přihlášení a je viditelný všem uživatelům. Ovšem jejich editaci, mazání a přidávání nových zpráv má ve své režii pouze administrátor.

6.2.9.1 Informační systém

Volba správy těchto novinek je umístěna v sekci Administrace. Administrátorovi se zobrazí seznam již přidaných novinek jak je vidět na obrázku 26. Jednotlivé záznamy může editovat a mazat. V případě tvorby nové zprávy se zobrazí formulář, který je na obrázku 27. Po vytvoření novinky je zobrazen její souhrn a poté je proveden návrat do výpisu všech novinek.

Obrázek 26: Tabulka novinek



Obrázek 27: Formulář novinky

6.2.9.2 Aplikace pro automatické řízení serveru

Přidávání a správa novinek je pouze v rámci části informačního systému, a proto žádná z akcí nepřijde do styku přímo se serverem. Aplikace pro automatické řízení serveru se do těchto akcí nezapojuje.

6.2.10 Nastavení

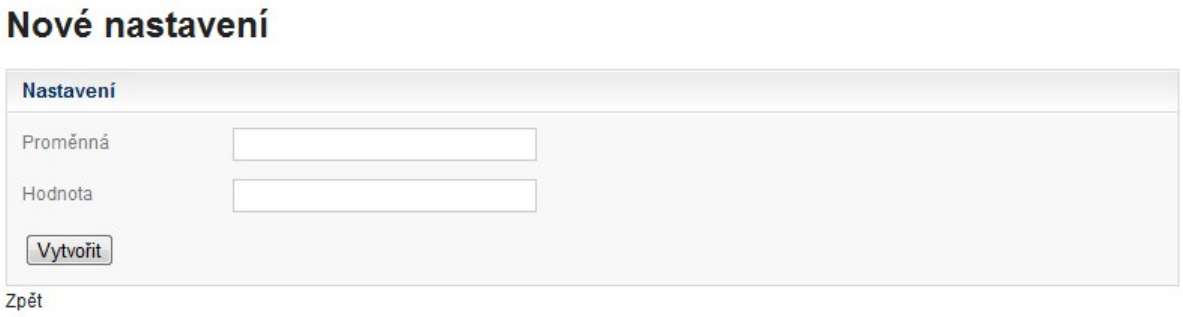
Volba nastavení je dostupná v sekci Administrace a má k ní přístup administrátor serveru. Pod touto volbou se skrývá tabulka s výpisem všech nastavení a direktiv, které informační systém potřebuje ke svému chodu. Je to například proměnná emailu odesílatele pro rozesílání emailových zpráv administrátorem, cesta k aplikaci daemona (aplikace pro automatické řízení serveru), apod.

6.2.10.1 Informační systém

V sekci Administrace se nachází položka Nastavení, kterou uživatel zvolí a zobrazí se mu tabulka s výpisem již přidaných nastavení systému. Ukázka této tabulky je vidět na obrázku 28.

Obrázek 28: Tabulka nastavení

Administrátor má možnost některý z těchto záznamů upravit nebo smazat. Možnost je vytvořit i novou direktivu, tato volba se skrývá pod odkazem Nové nastavení. Pokud administrátor toto nastavení bude chtít přidat, zobrazí se mu formulář, který je vidět na obrázku 29. Po vytvoření se vypíše hlášení a zobrazí detail právě vytvořeného nastavení.



Obrázek 29: Formulář přidání nastavení

6.2.10.2 Aplikace pro automatické řízení serveru

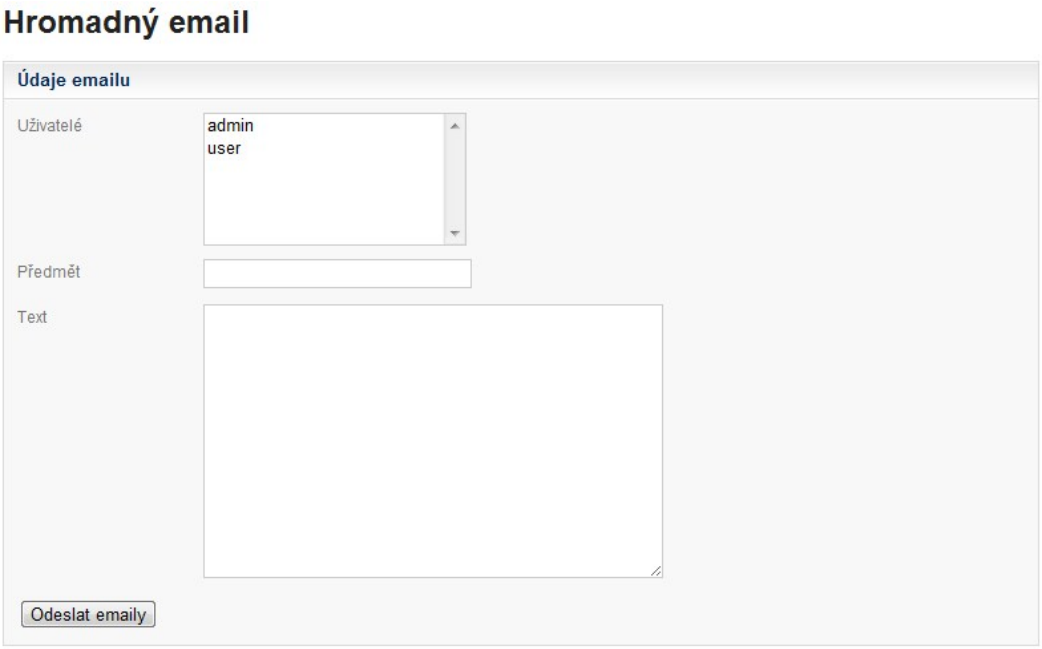
V případě této akce nebude zahájena žádná komunikace s aplikací pro automatické řízení serveru, neboť všechna nastavení budou ukládána a čtena z databáze informačního systému. Není potřeba se serverem provádět jakoukoliv činnost.

6.2.11 Hromadný email a log

Toto jsou doplňkové funkce pouze pro administrátora. Jedná se o zaslání informačních emailů zákazníkům serveru a možnost nahlédnutí do logu, kde jsou záznamy akcí, které byly se serverem uživateli prováděny. Log lze samozřejmě promazat, když bude obsahovat již mnoho záznamů.

6.2.11.1 Informační systém

V systému je pod volbou zaslání hromadných emailů zobrazen formulář, kde administrátor vybere uživatele (možno více najednou), kteří jsou načteni z databáze a po napsání emailu ho hromadně odešle. Tento formulář lze vidět na obrázku 30.



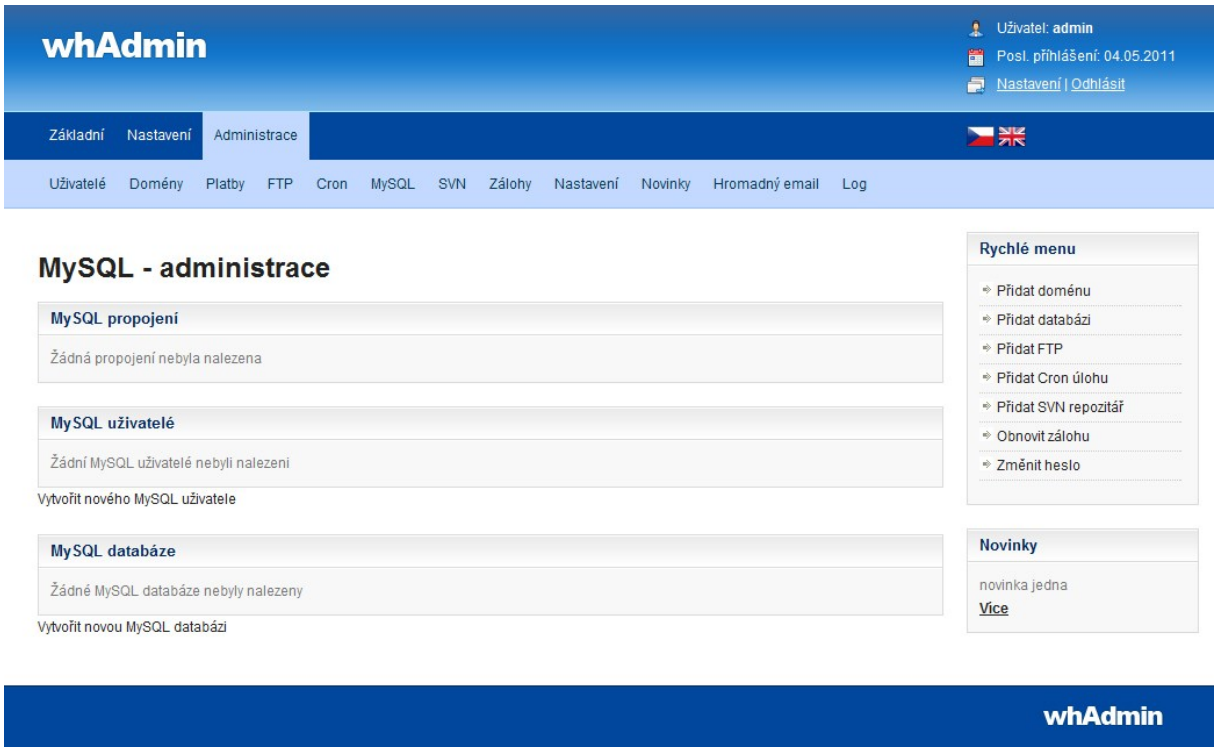
Obrázek 30: Formulář odeslání hromadného emailu

6.2.11.2 Aplikace pro automatické řízení serveru

V případě těchto funkcí nepřichází informační systém do styku s aplikací pro automatické řízení serveru, protože všechny údaje, ať už emaily zákazníků, nebo záznamy logu, dohledá systém ve své databázi.

6.3 Uživatelské rozhraní

V návrhu mělo uživatelské rozhraní pouze konceptuální formu. Na obrázku 31 je vidět obrazovka finální verze aplikace whAdmin.



Obrázek 31: whAdmin

7 Nasazení

Jednou z posledních etap vývoje informačního systému je jeho nasazení do běžného provozu. Před tím, než je ale aplikace do provozu uvedena, musí být řádně otestována a odladěna od nejzávažnějších systémových nebo bezpečnostních chyb.

7.1 Konfigurace serveru

Server, na kterém je výsledná aplikace nasazena, je virtualizovaný a má hardwarovou konfiguraci podle tabulky 5. Operační systém serveru je na bázi Linuxu, konkrétně se jedná o distribuci OpenSUSE 11.4. v 64 bitové verzi.

Procesor:	1x QEMU Virtual CPU, 2.33 GHz
Paměť RAM:	1024 MB RAM
Pevný disk:	jeden 1000 GB disk
Připojení:	1 Gbps

Tabulka 5: Konfigurace serveru

Část s informačním systémem je nasazena na HTTP server Mongrel[13], který je vhodný pro aplikace napsané ve frameworku Ruby on Rails. Ve vývojové části implementace bylo použito serveru Webrick. Na serveru musí být nainstalováno Ruby minimálně ve verzi 1.8.7.

7.2 Spouštění

Po nainstalování serveru Apache a zprovoznění služeb pro Cron, FTP, MySQL databázi, Ruby a SVN je postup následující.

7.2.1 Umístění aplikací na server

Na server se umístí aplikace pro vzdálený přístup ke správě serveru (informační systém whAdmin) a aplikace pro automatické řízení serveru (Daemon). V adresáři Daemona je nutné upravit konfigurační soubor conf.xml, kde se nastaví databáze whAdmina, přístup do ní pomocí jména a hesla, dále port pro XML-RPC server, nastavení pro adresáře, Apache a MySQL.

[13] http://rubyforge.org/projects/mongrel

7.2.2 Databáze

Z aplikace whAdmin je nutné migrovat databázové schéma do databáze a naplnit databázi počátečními daty. Provede se to spuštěním příkazu `bash ./db_recreate_production.sh`. Poté je nutné restartovat server příkazem `bash ./restart_server_production.sh`. Tímto se naplní databáze daty a je možné spustit systém whAdmin. Je vytvořen počáteční uživatel s právy administrátora s těmito údaji pro přihlášení:

- Login: admin
- Heslo: admin

7.2.3 Spuštění XML-RPC serveru

XML-RPC server se spustí pomocí příkazu `bash ./main.rb --server`. Tento soubor je umístěn v adresáři s aplikací Daemona. Poté bude server již připraven přijímat požadavky systému whAdmin.

7.2.4 Další nastavení

V informačním systému whAdmin je možno přenastavit některé volby v administračním menu ve volbě Nastavení. Jedná se hlavně o tyto volby:

- main_admin_email – hlavní email administrátora hostingu
- host – nastaveno na localhost
- port_daemon – HTTP port, na kterém naslouchá Daemon.

8 Závěr

Cílem této diplomové práce bylo vyvinout aplikaci pro vzdálenou správu hostingu. Mým úkolem bylo také zhodnotit stávající řešení, která jsou dostupná na internetu, určit jejich výhody a také jejich nedostatky. Většina z těch nejpopulárnějších aplikací, které byly na internetu volně k dostání měla nedostatky v tvorbě Cron úloh, nebo tuto službu neměly naimplementovanou vůbec. V některých aplikacích neměl administrátor možnost měnit všechny služby všem uživatelům, ale měl například na starosti pouze chod serveru a jeho nastavení. Mé řešení, které jsem nazval whAdmin, tyto nedostatky nemá.

Největším otazníkem nad implementací byla volba toho správného programovacího jazyka, který by dokázal obsloužit server tak, jak je požadováno a zároveň by bylo možno v něm vytvořit i webový informační systém. PHP by sice část s informačním systémem pokrylo, ale pro daemona je nevhodný. Zvoleno bylo Ruby, jelikož jsem se chtěl naučit jazyk, který je pro mě nový a odlišný od těch, které již znám. Ruby je vhodným jazykem i pro jeho MVC framework Ruby on Rails v němž byl napsán informační systém.

Můj systém whAdmin splnil požadavky zadání, poskytuje propracovanou správu uživatelských účtů, FTP uživatelů, správu domén, pro něž lze vytvořit subdomény a doménové aliasy. Mezi důležité funkce byla zahrnuta správa Cron úloh, které whAdmin také obsahuje. Dalšími funkcemi, které byly úspěšně naimplementovány jsou správa SVN repozitářů, jenž byly rovněž využívány při vývoji aplikace whAdmin, dále mezi nepostradatelné funkce patří správa databází a jejich uživatelů. Některé ze srovnávaných aplikací neobsahovaly správu záloh, což do systému whAdmin implementováno bylo a to jak správa webových záloh, tak i databázových. Samozřejmostí je možnost obnovy těchto záloh libovolně dle uvážení každého majitele svých cenných dat. Z časového důvodu a složitosti implementace nebyla realizována funkce VPN serveru. Databáze SQLite nebyla použita, protože je méně často využívána než MySQL, proto byla funkčnost realizována pro tuto databázi.

Pro budoucí možný vývoj tohoto systému pro vzdálenou správu hostingu je v plánu realizace funkcí pro správu email serverů, která je již v některých konkurenčních systémech obsažena, ale nebyla původně požadavkem při tvorbě této aplikace. Systém je možno rozšířit i jakoukoliv jinou funkcí, protože se jedná o volně dostupný produkt.

Seznam literatury

[1] ŠARMANOVÁ, Jana. *Teorie zpracování dat*. Ostrava : VŠB-TU Ostrava, 1997. 108 s. ISBN 80-7078-491-1.

[2] FULTON, Hal. *Ruby : kompendium znalostí pro začátečníky i profesionály*. Vyd. 1. Brno : Zoner Press, 2009. 765 s. ISBN 978-80-7413-018-2.

[3] *Apache* [online]. 2011 [cit. 2011-05-04]. Apache Subversion Documentation. Dostupné z WWW: <http://subversion.apache.org/docs/>.

[4] *The Open Group : crontab* [online]. 2001 [cit. 2011-05-04]. The Open Group Base Specifications Issue 7. Dostupné z WWW: <http://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html>.

[5] *Ruby on Rails* [online]. 2011 [cit. 2011-05-04]. Ruby on Rails Guides. Dostupné z WWW: <http://guides.rubyonrails.org/>.

[6] *DeveloperWorks* [online]. 2008 [cit. 2011-05-04]. RESTful Web services: The basics. Dostupné z WWW: <https://www.ibm.com/developerworks/webservices/library/ws-restful/>.

[7] *HTTP/1.1: Method Definitions* [online]. 1999 [cit. 2011-05-04]. Method Definitions. Dostupné z WWW: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>.

[8] *HTTP/1.1: Status Code Definitions* [online]. 1999 [cit. 2011-05-04]. Status Code Definitions. Dostupné z WWW: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2>.

[9] *Apache* [online]. 1996 [cit. 2011-05-04]. Apache HTTP Server. Dostupné z WWW: <http://httpd.apache.org/>.

Seznam příloh

Adresářová struktura přiloženého DVD

/Aplikace	Verze aplikace, která jde spustit (nainstalovat) na webovém serveru
/Prilohy	Adresář obsahuje přílohy k práci
/Prirucky	Uživatelská příručka a programátorská dokumentace k aplikaci
/Texty	Soubory s textem práce, zadání, klíčová slova a abstrakt